UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Šircelj

Razvoj mobilne aplikacije za pomoč študentom pri organizaciji študija

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Matej Šircelj

Razvoj mobilne aplikacije za pomoč študentom pri organizaciji študija

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Tomaž Hovelja

SOMENTOR: doc. dr. Rok Rupnik

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Kandidat naj analizira trg mobilnih aplikacij za pomoč študentom pri študiju. Na podlagi analize naj določi nabor funkcij, ki jih taka aplikacija mora imeti, nabor funkcij, ki so za aplikacijo pomembne, niso pa nujne ter nabor funkcij, ki so v aplikaciji dobrodošle. Kandidat naj tudi pregleda orodja za razvoj mobilnih aplikacij in izbere tehnično najprimernejša orodja za razvoj aplikacije za pomoč študentom pri študiju. Nato naj kandidat razvije prototip take aplikacije, ki bo prilagojena potrebam študija na Fakulteti za računalništvo in informatiko.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Matej Šircelj sem avtor diplomskega dela z naslovom:

Razvoj mobilne aplikacije za pomoč študentom pri organizaciji študija

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Tomaža Hovelje in somentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 19. februarja 2016 Podpis avtorja:

Zahvaljujem se Martini Šircelj za lektoriranje diplomske naloge.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne stilske podloge
AWS	Amazon Web Services	Amazonove Spletne Storitve
AMI	Amazon Machine Image	Amazonova strojna slika
MD5	message digest 5	kriptografska zgoščevalna funkcija
IE	Internet Explorer	Internet Explorer
DOM	document object model	model objektov dokumenta
CMD	command	ukaz
IDE	integrated development environment	integrirano razvojno okolje
URL	uniform resource locator	enotni naslov vira
CDN	content delivery network	omrežje za dostavo vsebin
DNS	domain name system	sistem domenskih imen
DSDM	Driving Strategy Delivering More	konzorcij Driving Strategy Delivering More
MUS	minimal usable subset	najmanjša uporabna podmnožica
VCS	version control system	sistem za nadzor različic

Povzetek

V diplomski nalogi sta opisana motivacija in potek razvoja mobilne aplikacije za pomoč študentom pri organizaciji študija. V sklopu diplomskega dela je bil tudi razvit prototip aplikacije za mobilna operacijska sistema Android in iOS. Prototip omogoča študentom in osebju dostop do interaktivnega urnika, sporočil, indeksa in splošnih informacij o fakulteti in osebju. Hkrati tudi omogoča komunikacijo med pedagogi in študenti.

Izdelana je bila analiza trga, v kateri smo pregledali podobne aplikacije. Informacije, ki smo jih dobili z našo analizo trga, smo uporabili pri zasnovi in razvoju naše aplikacije. Na podlagi teh ugotovitev smo določili katere funkcionalnosti so pomembne in izločili nepotrebne ali celo nezaželene funkcionalnosti.

Aplikacija je bila narejena na podlagi tehnologije HTML5 (HTML, CSS, javascript). Za potrebe komunikacije med aplikacijo in strežniki je bilo izdelano tudi zaledje z uporabo Amazonovih storitev v oblaku »Amazon Web Services (AWS)«. Zaledni strežnik je bil razvit z uporabo knjižnice node.js, in je prav tako napisan v jeziku javascript.

Ključne besede: mobilna aplikacija, pomoč pri študiju

Abstract

In this thesis we described the motivation for and development of a mobile application for study organising. The result of the thesis is amobile application prototype for Android and iOS. This prototype allows students and staff access to interactive timetable, messages, study index and general information about the faculty and staff. It also serves as a platform for communication between staff and students.

To define key developed functionalities of the prototypewe conducted a market analysis in which we analysed similar mobile applications. Based on the information gathered through the market analysis we determined which functionalities are important and which are unnecessary.

The application was made with HTML5 technologies (HTML, CSS, javasript). A backend service was made on Amazon AWS platform for communication purposes between the mobile application and server. The backend server was developed on node.js software stack which is also written in javascript

Keywords: mobile application, help with studying

Poglavje 1 Uvod

Študiji postajajo vedno bolj časovno intenzivni, zato zmanjšanje časovnega vložka pri stranskih procesih omogoča študentom, da se lahko bolj osredotočijo na samo učenje. Pedagogi pa se lahko osredotočijo na kvaliteto predavanj oz. vaj. Hkrati lahko izbolšanje administrativni procesov izboljša komunikacijo med študenti in pedagogi. Glede na napovedi o vse večji uporabi mobilnih naprav, tako v Sloveniji kot tudi v svetovnem merilu, lahko mobilna aplikacija, ki poizkuša nasloviti zgoraj opisano problematiko prispeva k izboljšanju produktivnosti pri izvedbi študija tako študentom kot pedagogom.

Namen diplomske naloge je najprej identificirati potencialne koristi mobilne aplikacije za pomoč študentom pri organizaciji študija. Nato pa razviti prototip, ki bo vseboval funkcije, funkcije (koledar, sporočila, indeks, …) s katerimi je te koristi možno doseči .

Motivacija za moje delo je olajšati študentom organizacijo študija. Menim, da bi mobilna aplikacija FRI študentom in pedagogom olajšala administrativne procese in povečala uporabo trenutnih študijskih informacijskih sistemov (eUčilnica in eŠtudent) preko mobilnih naprav. Tako bi lahko fakulteta svoje informacijske sisteme razširila še na mobilne uporabnike.

Da bi tako mobilno aplikacijo lahko razvili smo siv diplomski nalogi zastavili tri cilje:

1. analizirati trg in določiti nabor funkcij, ki jih je potrebno razviti,
2. pregledati orodja za razvoj aplikacije in izbrati tehnično najprimernejše,
3. razviti aplikacijo

Za dosego prvega cilja bomo v naslednjem poglavju analizirali trg mobilnih aplikacij in pregledali tako generične aplikacije kot tudi specifične že obstoječe aplikacije posameznih fakultet. Kljub temu, da smo se odločili razviti mobilno aplikacijo specifično za FRI, je smiselno vključiti v analizo tudi generične aplikacije. Generične aplikacije imajo v osnovi veliko večji trg in hkrati tudi večjo konkurenco na trgu. V analizi trga smo ugotovili, da so večinoma bolje izdelane in imajo več uporabnikov.

Pri analizi trga smo ugotovili, da je mnogo generičnih aplikacij, ki so bile pri uporabnikih dobro sprejete in so dosegle veliko število uporabnikov. Pri specifičnih aplikacijah pa je delež univerz oz. fakultet, ki sploh imajo takšne aplikacije relativno majhen. V naši analizi bomo poskusili odgovoriti na vprašanje: »Kaj je vzrok, da na trgu primanjkuje specifičnih aplikacij za pomoč pri študiju?«. Glede tega vprašanja smo si zastavili dodatna podvprašanja: ali je vzrok v tem, da ni zanimanja za tako vrsto aplikacije? (slaba ideja), ali je vzrok v tem, da ni zaupanja v tako vrsto aplikacije? (čas ni pravi), ali je vzrok v tem, da aplikacije niso dovolj dobre? (slaba izvedba)Pri izboru orodij in izdelavi aplikacije smo posvetili posebno pozornost prav tem podvprašanjem. Za dobro trženje produkta, pa naj bo ta zastonj ali plačljiv, je namreč potrebno poskrbeti, da je ideja prava, dobro izvedena in poslana na trg ob pravem času.

S specifično aplikacijo se da lažje doseči velik odstotek uporabnikov, saj lahko postane obvezno orodje v študijskem procesu. Na primer uporaba eUčilnice na FRI je za študente obvezna. Ko se zahteva obvezna uporaba (ang. mandatory use) je pomembno paziti na zagotavljanje enakih možnosti vseh študentov in tistim, ki nimajo dostopa do aplikacije (nimajo telefona, telefon ne podpira aplikacije, …) omogočiti dostop preko drugih poti.

S stališča samega uporabniškega vmesnika se generična in specifična vrsta aplikacije le malo razlikujeta. Razlika nastane predvsem, pri vsebini in umestitvi le-te v uporabniški vmesnik. Komunikacija in vsebine, ki nastanejo v trenutnem informacijskem sistemu so zelo pomembni saj brez tega še tako dober uporabniški vmesnik ne pomeni nič. Tega se je treba zavedati in aplikacijo razviti tako, da služi predvsem namenu izboljšanja dostopnosti teh elementov. Treba se je tudi opredeliti glede tega, katere funkcionalnosti aplikacije bodo integrirane z obstoječim informacijskim sistemom fakultete.

Za dosego drugega cilja se bomo v tretjem poglavju diplomske naloge predvsem osredotočili na vprašanje, katera razvojna orodja lahko omogočijo uporabo aplikacije čim več uporabnikom. Tukaj je ključno vprašanje, kako zastaviti razvoj za Android in iOS. Če že ne v izgledu, pa mora biti aplikacija vsaj identična po funkcionalnosti na vseh mobilnih operacijskih sistemih.

Android in iOS temeljita na različnih programskih jezikih. Android je napisan v Javi in iOS v C#. Za razvoj uporabljata tudi popolnoma drugačna orodja in operacijske sisteme. Za iOS je namreč potrebno za pretvorbo kode aplikacije v strojni jezik (compile) imeti Applov namizni operacijski sistem in posledično tudi Applov računalnik. Te razlike nam lahko predstavljajo problem, ki ga je potrebno razrešiti še pred začetkom razvoja.

Za razvoj aplikacije smo se odločili uporabiti tehnologije HTML5 in programsko knjižnico Cordova. Tej odločitvi so botrovale štiri stvari: o teh tehnologijah imam veliko znanja, uporaba teh tehnologij omogoča poenoten razvoj za obe platformi v večini postopkov na poti do končne aplikacije, tehnologije omogočajo poenoten uporabniški vmesnik na obeh platformah, tehnologije omogočajo poenoteno vzdrževanje aplikacije

Pri uresničevanju tretjega cilja, razvoj prototipa aplikacije, pa smo se v četrtem poglavju osredotočil predvsem na to, kako čim bolje izkoristiti izbrane tehnologije pri uporabniškem vmesniku, procesih komunikacije med aplikacijo in strežnikom in hranjenjem podatkov v mobilnih napravah.

Poglavje 2 Analiza trga

Z analizo trga bomo uresničili prvi cilj tega diplomskega dela. V tem poglavju bomo poizkušali odgovoriti na vprašanja o velikosti potencialnega kroga uporabnikov (trg), o potrebi študentov za uporabo takšne vrste aplikacije. Dodatna vprašanja, ki nas bodo zanimala so: kaj bo naša konkurenčna prednost, kako se bomo branili pred posnemanjem našega produkta s strani konkurence in koliko uporabnikov mobilne aplikacije lahko pričakujemo in ali se fakultetam oz. univerzam izplača investirati v takšno mobilno aplikacijo. Odgovori na ta vprašanja bodo vplivali na načrtovanje aplikacije, ki ga bomo opisali v drugem poglavju.

2.1 Pregled trga

Pri pregledu trga smo se najprej osredotočili na pregled najbolj uporabljanih šolskih informacijskih sistemov [1]. Zanimalo nas je kakšne mobilne rešitve nudijo kot dodatek k njihovim platformam, ki delujejo na osebnih računalnikih. Obravnavali smo 4 največje ponudnike na ameriškem trgu:

- Blackboard Learn (35,4% tržnega deleža)
- Moodle (20.6% tržnega deleža)
- Canvas (15.2% tržnega deleža)
- Desire2Learn Brightspace (9.1% tržnega deleža)

Izmed navedenih študijskih informacijskih sistemov je le Moodle odprtokodni projekt. Pri tem ponudniku imamo odprte roke pri integraciji. Pri drugih pa se moramo zanašati na njihove API-je, če ti seveda obstajajo.

Pri ponudniku Blackboard Learn je na voljo le ena mobilna aplikacija [2] za povezavo na katerikoli njihov šolski sistem. To jasno kaže na to, da Blackboard Learn ne omogoča povezave z njihovim študijskim sistemom zunanjim izvajalcem.

Canvas in Desire2Learn Brightspace nudita partnerski program, ki zunanjim izvajalcem omogoča razvoj svojih orodij. Canvas ima za ta namen svoj partnerski portal »eduappcenter« [3]. Desire2Learn Brightspace pa ima portal Brightspace App Finder [4].

Ponudnik mobilnih aplikacij za platformo BrightSpace, DubLabs, ima na svoji spletni strani [5] navedenih preko 40 univerz in K-12 osnovnih šol, ki uporabljajo njihovo mobilno aplikacijo. Njihove aplikacije so dostopne tako na Google Play kot iTunes AppStore.

Ponudniki specifičnih aplikacij so pri večini izbranih aplikacij kar same univerze. Iz podatkov, ki so nam dostopni iz Google Play in iTunes AppStore trgovin, ne moremo narediti konkretnih sklepov o ponudnikih, saj so aplikacije verjetno naredili podizvajalci. Če se osredotočimo na same aplikacije in njihov izgled pa lahko ugotovimo, da se vse razen treh popolnoma razlikujejo.

Iz teh podatkov lahko ugotavljamo, kje na črti me monopolnim trgom in popolno konkurenco je trg študijskih informacijskih sistemov. Popolna konkurenca pomeni takšno nasičenost trga s konkurenčnimi si rešitvami, da so dobički tako majhni, da lahko vsi igralci le preživijo. V primeru popolnega monopola pa je na trgu le en igralec, ki nadzira celoten trg in je novim podjetjem praktično onemogočen vstop na trg [6].

Treba je obrazložiti, da sta študijski informacijski sistemi in mobilne aplikacije dva različna segmenta pri obravnavi in, da trenutno obravnavamo študijske informacijske sisteme. Bralcu se morda zato na prvi pogled zdi nesmiselno obravnavati trg študijskih informacijskih sistemov saj s samo mobilno aplikacijo ne mislimo vstopati v točno ta določen trg. Tukaj je treba povedati, da smo kot ustvarjalci mobilne aplikacije odvisni od študijskih informacijskih sistemov in da jih moramo obravnavati kot poslovne partnerje. Če se kot razvijalci odločimo razviti aplikacijo za določen študijski informacijski sistem, želimo biti seznanjeni s temi informacijami in razviti aplikacijo v skladu z njihovo vizijo.

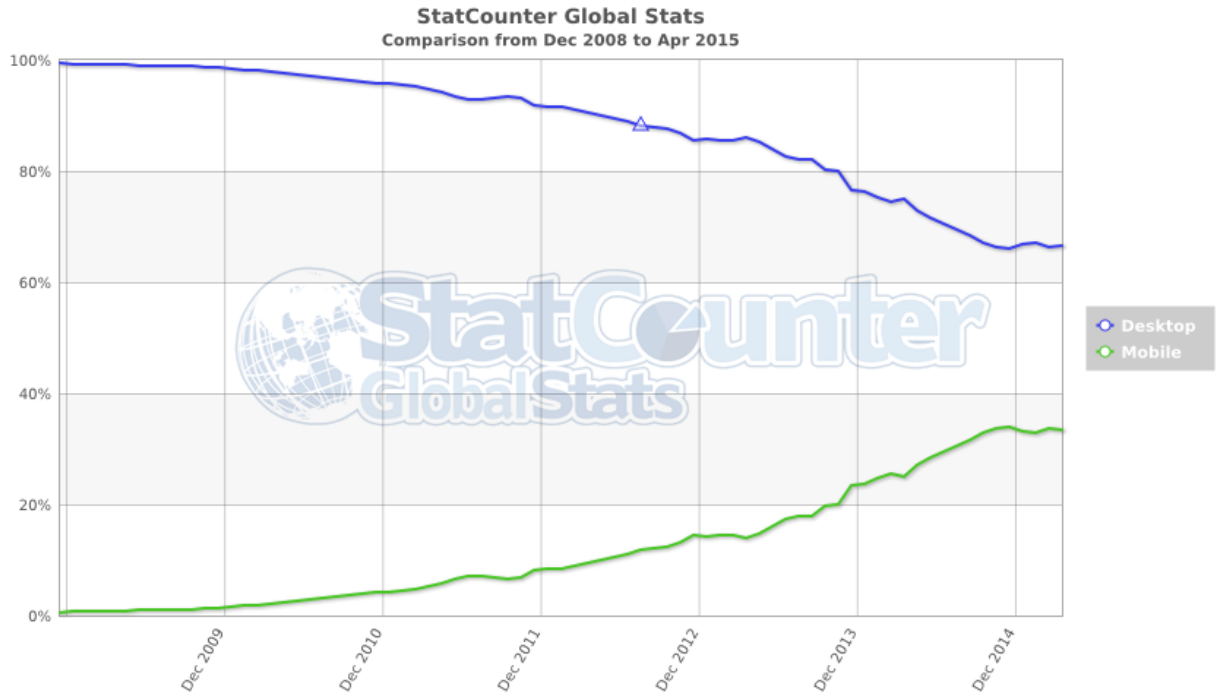
Ne moremo trditi, da je trg monopolen, saj imamo vsaj 4 dokaj enakovredne igralce. Poleg tega Blackboard Learn, trenutno še vedno največji igralec na ameriškem trgu, izgublja tržni delež [7]. Ne moremo tudi trditi, da je trg popolna konkurenca zaradi premajhnega števila igralcev in prevelike skoncentriranosti tržnega deleža pri določenih igralcih.

V tem primeru se je pametno osredotočiti na določeno nišo kjer mislimo, da imamo konkurenčno prednost in v primerjavi s konkurenco lahko strankam ponudimo dodano vrednost oziroma smo boljši. V našem primeru bi se tako lahko osredotočili na interakcijo med študenti in pedagogi. Medtem, ko ostale mobilne aplikacije omogočajo opravljanje osnovnih

administrativnih nalog in nudijo študentom le enosmerne informacije, bi lahko naša aplikacija omogočila študentom in pedagogom boljšo komunikacijo.

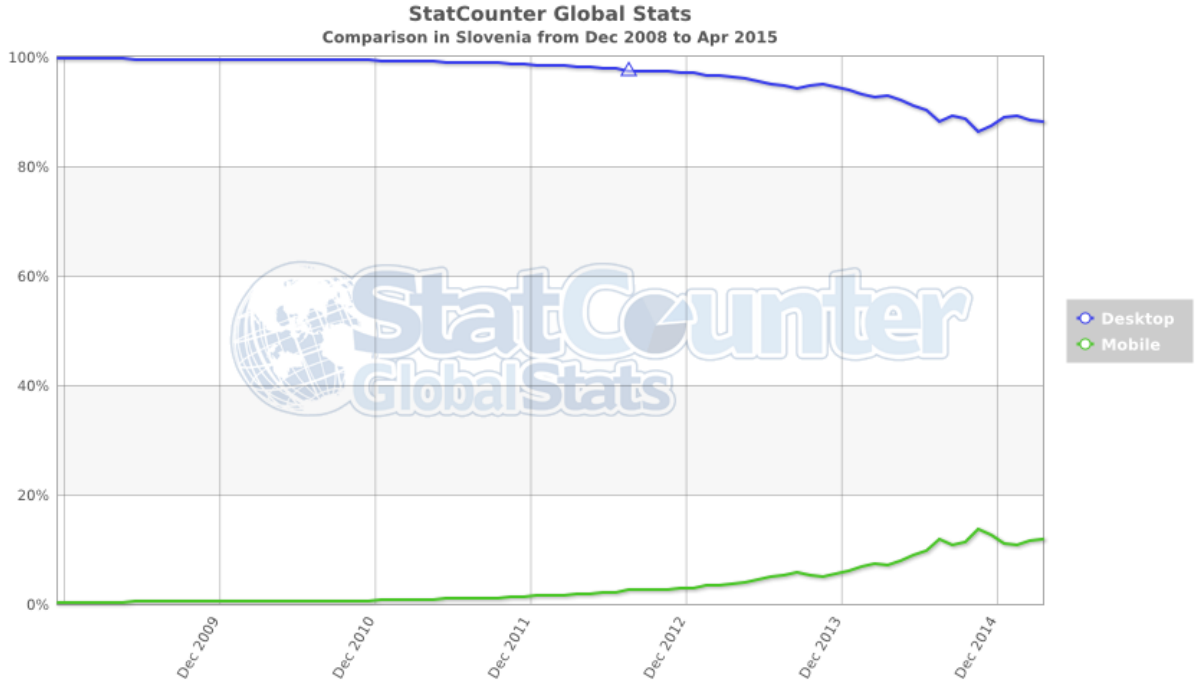
Po pregledu trga uporabljanih šolskin informacijski sistemov bomo preučili še trg mobilnih naprav in mobilnih operacijskih sistemov. Analiza tega trga je za nas pomembna, ker se moramo odločiti, katere mobilne operacijske sisteme bomo podprli.

Aprila 2015 je globalni obisk spletnih strani preko mobilnih naprav po podatkih organizacije StatCounter Global Stats dosegel 33.5% obiska iz stacionarnih računalnikov. To je v primerjavi z aprilom 2014 porast za 8.5 odstotnih točk, ko je bil obisk 25%. Podatki zadnjih let kažejo na to, da se uporaba mobilnih naprav v zadnjih letih povečuje približno za 9 odstotnih točk na leto.

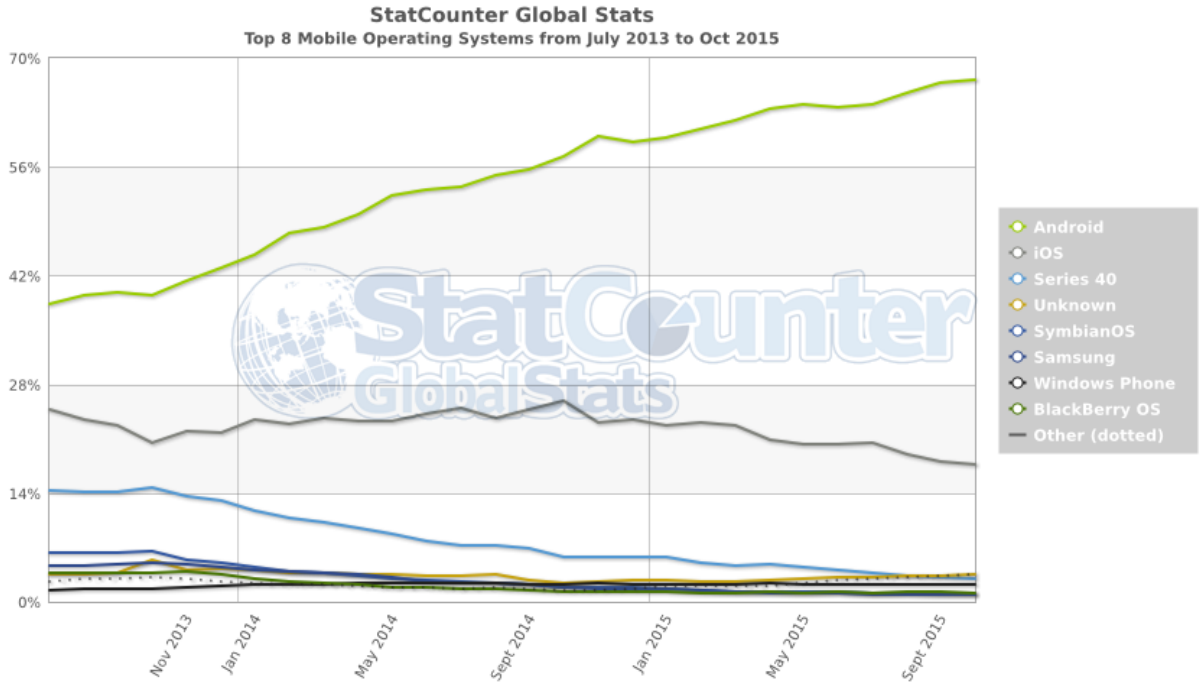


Slika 1.1: Svetovna statistika obiska spletnih strani preko mobilnih naprav in osebnih računalnikov

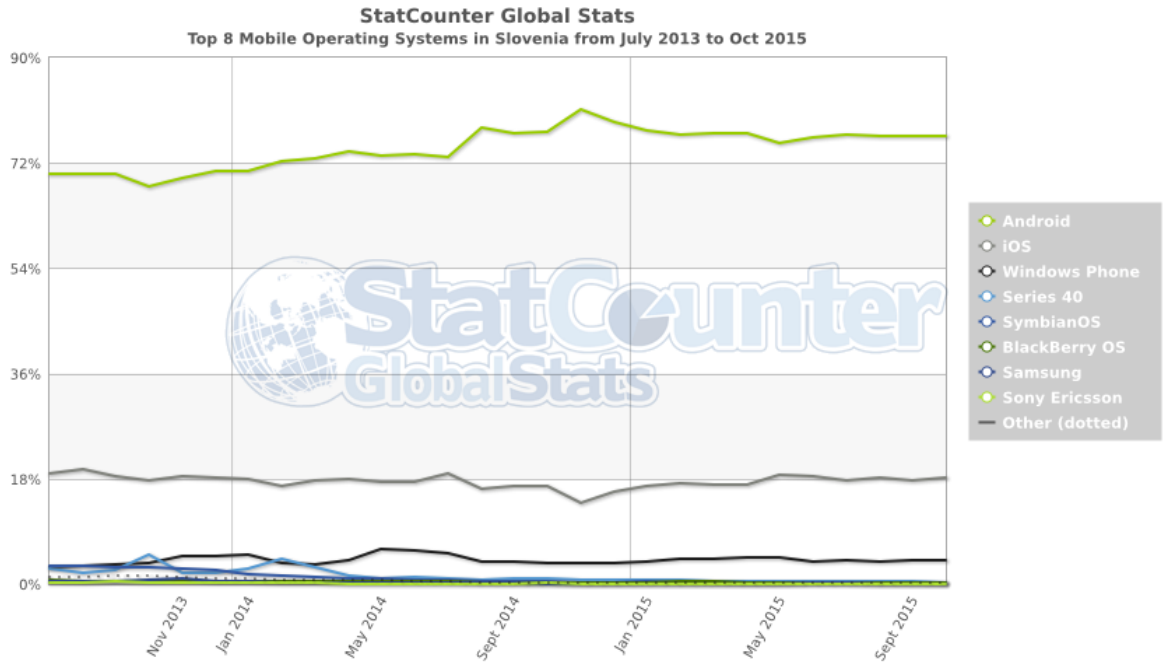
V Sloveniji je bil odstotek prenosov v aprilu 2015 okoli 14% in se je v primerjavi z aprilom 2014 povečal za 7 odstotnih točk.



Slika 1.2: Slovenska statistika obiska spletnih strani preko mobilnih naprav in osebnih računalnikov.



Slika 1.3: Svetovna statistika operacijskih sistemov na mobilnih napravah



Slika 1.4: Slovenska statistika operacijskih sistemov na mobilnih napravah

Pri pregledu trga so bile obravnavane aplikacije iz trgovin Google Play in iTunes App store. Operacijska sistema Android in iOS sta imela v drugem četrtletju 2015 96.7% delež na trgu pametnih telefonov [8]. V prvi sklop so bile izbrane specifične Android in iOS aplikacije. V drugi sklop pa generične aplikacije.

Da je specifična aplikacija prišla v izbor, je morala izpolnjevati naslednje kriterije:

- Aplikacija mora uporabnikom omogočati pregled nad dejavnostmi, ki se izvajajo na univerzi.
- Aplikacija mora biti v angleškem ali slovenskem jeziku.
- Aplikacija mora biti povezana z informacijskim sistemom univerze ali fakultete.

Za iskalni niz je bil v primeru specifičnih aplikacij uporabljena beseda »university«. Rezultati so bili v obeh trgovinah dovolj izčrpni, da iskanje z drugimi nizi ni našlo dodatnih uporabnih aplikacij. Iz množice rezultatov so kriterijem za obravnavo zadostovale naslednje:

2.1.1 Število namestitev in ocene specifičnih aplikacij

Aplikacija	*Število namestitev*	*Število ocen*	*Ocena*
***University of Phoenix** [9]*	500,000 - 1,000,000	7,266	4.1 / 5
***Ashford University Mobile** [10]*	100,000 - 500,000	1,780	4.3 / 5
***Christ University Student App** [11]*	10,000 - 50,000	515	4.2 / 5
***Monash University** [12]*	10,000 - 50,000	411	3.8 / 5
***Kingston University** [13]*	10,000 - 50,000	113	4.1 / 5
***Newcastle University** [14]*	10,000 - 50,000	130	4.5 / 5
***University Bordeaux Schedule** [15]*	1,000 - 5,000	118	4.7 / 5
***Monroe Community College** [16]*	5,000 - 10,000	115	3.9 / 5
***Calvin College** [17]*	1,000 - 5,000	14	4.6 / 5
***Hathershaw College** [18]*	100 - 500	13	1.9
***Emerson College** [19]*	100 - 500	3	5.0
***St Charles Sixth Form College** [20]*	50 – 100	3	4.7
***Visit Fisher** [21]*	50 - 100	0	/

Trgovina z aplikacijami iTunes App Store ne prikazuje podatkov o prenosih. Ocene uporabnikov so dostopne vendar so segmentirane po državah. Tako ne moremo enostavno dobiti lestvice najbolje ocenjenih ali uporabljanih aplikacij. Pri izboru iOS aplikacij smo si tako pomagali z zunanjim orodjem SensorTower [22], ki hrani podatke o ocenah in pozicijah aplikacij v »iTunes App Store« trgovini.

Aplikacija	*Št. Ocen*	*Povprečna ocena*	*Primarna država*
***UniOnline for KF Graz** [23]*	13	4	Avstrija
***OOHLALA - Campus App** [24]*	281	3.75	ZDA + Kanada
***iWestminster** [25]*	42	3	Velika Britanija
***University of Texas At Austin** [26]*	3.535	4.0	ZDA
***University of Phoenix Mobile** [27]*	1,338	2.5	ZDA
***University of Sussex** [28]*	38	4	ZDA

Univerze ki so na vrhu lestvice, imajo tako veliko število prenosov, ker so to t.i. »online« univerze in imajo veliko število študentov, ki sploh ne obiskujejo kampusa. University of Phoenix ima tako okoli 300.000 vpisanih študentov na leto. Iz te lestvice lahko tako ugotovimo, da na sprejem in uporabljanost aplikacije predvsem vplivajo kvaliteta aplikacije, ali je mobilna aplikacija primaren ali sekundaren način opravljanja administrativnih nalog in velikost skupine uporabnikov

2.1.2 Število namestitev in ocene generičnih aplikacij

Aplikacija	*Število namestitev*	*Število ocen*	*Ocena*
***Timetable** [29]*	1,000,000 - 5,000,000	35,034	4.3 / 5
***My Study Life** [30]*	500,000 – 1,000,000	26,722	4.2 / 5
***Google Classroom** [31]*	1,000,000 – 5,000,000	24,659	4.0 / 5
***myHomework Student Planner** [32]*	1,000,000 – 5,000,000	24,102	4.0 / 5
***Student Agenda** [33]*	500,000 – 1,000,000	21,120	4.2 / 5
***Handy Timetable** [34]*	1,000,000 – 5,000,000	20,584	4.1 / 5
***My Class Schedule: Timetable** [35]*	500,000 – 1,000,000	8,879	4.1 / 5
***Students - Timetable** [36]*	50,000 - 100,000	1,289	4.2 / 5

Tudi pri pridobivanju podatkov o generičnih aplikacijah smo si pomagali z orodjem SensorTower.

Aplikacija	*Št. ocen*	*Povprečna ocena*	*Primarna država*
***myHomework Student Planner** [37]*	52,694	3,5	ZDA
***Pocket Schedule** [38]*	485	4	ZDA
***ClassManager** [39]*	383	4.5	ZDA
***Courses** [40]*	29	2.5	ZDA
***Planneroo™** [41]*	6	3.5	ZDA
***Timetable** [42]*	112	4.5	Rusija
***Homework Suite** [43]*	18	4.5	ZDA

Pri generičnih aplikacijah je opaziti veliko večjo izenačenost med prenosi. Kar 7 aplikacij v trgovini Google Play ima primerljivo število prenosov, število ocen in tudi oceno uporabnikov. To nam pove, da je tukaj na trgu zelo močna in izenačena konkurenca. Tako je smotrno preučiti te aplikacije s stališča kvalitete uporabniškega vmesnika in stremeti k enaki kakovosti.

2.2 Analiza funkcionalnosti

Pri ugotavljanju funkcionalnosti smo glede vsake opredelili po MoSCoW metodi [44].

Pri uporabi MoSCoW metode porazdelimo možne funkcionalnosti v naslednje skupine:

- Funkcionalnosti, ki jih aplikacija mora imeti (must have)
- Funkcionalnosti, ki so za aplikacijo pomembne, niso pa nujne (should have)
- Funkcionalnosti, ki so v aplikaciji dobrodošle, niso pa nujne (could have)
- Funkcionalnosti, ki niso primerne za aplikacijo (won't have)

Zgoraj naštete aplikacije smo pregledali in njihove funkcionalnosti razporedili v skupine. Za vsako funkcionalnost smo tudi zabeležili koliko aplikacij jo uporablja. Spodaj so naštete vse te funkcionalnosti in število aplikacij, ki jih uporabljajo.

- **Specifične aplikacije**

Funkcionalnost	***Število aplikacij, ki uporabljajo to funkcionalnost***	***Odstotek aplikacij, ki uporabljajo to funkcionalnost***
Urnik	15 / 19	79 %
Novice	10 / 19	53 %
Karta	10 / 19	53 %
Pregled osebja	7 / 19	37 %
Knjižnica	6 / 19	32 %
Osebna sporočila	5 / 19	26 %
Informacije	4 / 19	21 %
Indeks / Ocene	4 / 19	21 %
Forum / Razprave	3 / 19	16 %
Predmetnik	3 / 19	16 %
Avtobus	3 / 19	16 %
Zaposlitve	2 / 19	11 %
Prisotnost	1 / 19	5 %
Sindikat	1 / 19	5 %
Izpiti	1 / 19	5 %
Ekskurzije	1 / 19	5 %

Zunanje povezave	1 / 19	5 %

- **Generične aplikacije**

Funkcionalnost	***Število aplikacij, ki uporabljajo to funkcionalnost***	***Odstotek aplikacij, ki uporabljajo to funkcionalnost***
Urnik	13 / 15	87 %
Predmetnik	10 / 15	67 %
Naloge	4 / 15	26 %
Opomnik	3 / 15	20 %
Ocene	2 / 15	13 %

Da smo uvrstili zgornje funkcionalnosti v MoSCoW kategorije, smo se uprli na DSDM (konzorcij »Driving Strategy Delivering More«) smernice [45]. Aplikacije smo obravnavali skupaj, tako specifičen kot tudi generične.

- **Nujne funkcionalnosti (Must have)**

Koledar / Urnik
Novice / Sporočila

Nujne funkcionalnosti, so tiste, s katerimi dosežemo najmanjši uporabni izdelek (MUS – minimal usable subset). Brez realizacije teh funkcionalnosti je produkt neuporaben. Moramo se vprašati, kakšna bi bila aplikacija brez urnika in novic. Obe funkcionalnosti, omogočata komunikacijo med osebjem in študenti. Glede na to, da smo se odločili, da bo naša tržna niša prav poudarek na tej komunikaciji, bi ne implementacija teh funkcionalnosti pomenila popoln odmik od zastavljenih ciljev.

- **Pomembne funkcionalnosti (Should have)**

Pregled osebja
Informacije
Indeks / Ocene + Izpiti

To so funkcionalnosti, ki so pomembne vendar pa niso vitalne za delovanje naše aplikacije. Možno jih je izpustiti iz končnega produkta vendar pomankanje teh funkcionalnosti močno okrni njegovo vrednost. Tukaj moramo sami določiti kakšno težo imajo te funkcionalnosti saj je to edina razlika med pomembnimi in dobrodošlimi funkcionalnostmi (»should have« in »could have«). Pregled osebja je pomemben, ker ga vključuje tudi velika količina drugih aplikacij. Hkrati pa v aplikacijo vključimo informacije, ki zelo dopolnjujejo »must have« funkcionalnosti urnik in sporočila. Splošne informacije predstavljajo funkcionalnost, pri kateri se lahko sami odločimo katere najpomembnejše informacije o fakulteti bomo vključili. Indeks in prijava na izpite pa je funkcionalnost, ki bi lahko poenostavila prijavo na izpite in študentom omogočila veliko lažjo seznanitev s tem kateri izpiti se bližajo.

- **Dobrodošle funkcionalnosti (Could have)**

Karta
Knjižnica
Osebna sporočila
Forum / Razprave
Predmetnik
Osebna sporočila
Avtobus
Zaposlitve

To so funkcionalnosti, ki bi jih lahko vključili ampak ne prinašajo neke večje dodane vrednosti. Nekatere funkcionalnosti, kot so »karta« in »avtobus« so v tej kategorije zaradi specifike FRI in bi lahko v drugih primerih padle pod kategorijo pomembnih funkcionalnosti. V primeru, da bi v naši aplikaciji obravnavali celotno univerzo, bi ti dve funkcionalnosti prinesli veliko večjo dodano vrednost.

- **Neprimerne funkcionalnosti (Wont have)**

Prisotnost
Sindikat
Ekskurzije
Zunanje povezave

To so funkcionalnosti, ki so v našem produktu nezaželene. Pomembno je, da je naša aplikacija osredotočena na pomembne naloge in ne poskuša biti rešitev za vse. Zgornje funkcionalnosti bi prinesle več slabosti kot koristi v smislu procesne in računalniške kompleksnosti, zato smo jih uvrstili pod kategorijo neprimernih funkcionalnosti.

2.3 Analiza uporabnikov

Ko analiziramo uporabnike jih moramo ločiti na dva segmenta: fakultete, ki bodo aplikacijo uporabljale in študente in osebje na teh fakultetah. Velja namreč, da ti dve skupini nimata enakih potreb in bosta tudi uporabljali aplikacijo v popolnoma različne namene.

2.3.1 Študentje / Pedagogi

Čeprav bodo študentje in pedagogi uporabljali isto aplikacijo, jih ne moremo obravnavati kot eno skupino. Ena od ključnih funkcionalnosti je omogočanje komunikacije med študenti in pedagogi. Če ena skupina nebi bila motivirana za uporabo aplikacije, potem bi to pustilo posledice tudi na drugo skupino.

Pri tem so pedagogi, tisti ki ustvarjajo in moderirajo razpravo pri svojih predmetih. Za to morajo motivirati študente in hkrati dobivati dober odziv, da bo razprava potekala skozi celo šolsko leto. Tukaj se moramo vprašati, kako lahko aplikacija pedagogom nudi vpogled v uspešnost njihovih prizadevanj.

Na drugi strani so študentje, ki se morajo ob uporabi aplikacije počutiti dovolj udobno, da bo število uporabnikov pri določenem predmetu doseglo kritično maso. Hkrati mora aplikacija zagotavljati boljšo uporabniško izkušnjo v primerjavi s spletnim portalom šolskega informacijskega sistema. Ne samo na mobilnih napravah, ampak se je pomembno primerjati z uporabo na osebnih računalnikih. Pametno je v tem aspektu primerjati porabljen čas, učinkovitost in zadovoljstvo uporabnikov.

Klasično opredeljevanje glede spola in starosti uporabnikov je v našem primeru nepotrebno, saj se je skupina uporabnikov zelo homogenizirana. Razlika med spoloma je odvisna od posamezne fakultete. Če gledamo na domet uporabe aplikacije širše, na primer s stališča celotne univerze, pa je razdeljenost polovica moški in polovica ženske. Starostna skupina je enaka kot starostna skupina študentov in je večinoma med 18 in 30 let.

Bolj pomemnbno je določiti uporabniške skupine glede na to kakšen odnos imajo uporabniki do pametnih telefonov, mobilnih aplikacij in tehnologije na splošno. Tukaj lahko naletimo na 4 skupine uporabnikov katere bi lahko imele težave z uporabo aplikacije .

2.3.1.1 Ne dovolj obveščeni uporabniki

To so ljudje, ki sploh ne vedo, da naša mobilna aplikacija obstaja. Študentje morajo biti vključeni v administrativne procese fakultete bodisi preko sistema eŠtudent, bodisi preko eUčinlice. Tako tukaj ni večjih dvomov, da bodo vsi seznanjeni z aplikacijo. Večji poudarek je tukaj potrebno dati obveščanju osebja saj je več možnosti, da ne bodo obveščeni ali pa bodo ignorirali obvestila.

2.3.1.2 Tehnični analfabeti

To so ljudje, ki zavračajo tehnologijo in tehnične naprave, se jih bojijo ali nimajo znanja in ga tudi nočejo pridobiti. Na računalniških fakultetah izmed vseh ostalih fakultet verjetno najdemo najmanj takšnih ljudi, tako da se nam takih uporabnikov v našem primeru ni treba bati.

2.3.1.3 Travmatizirani uporabniki

To so ljudje, ki so imeli v preteklosti slabe izkušnje z mobilnimi aplikacijami ali celo z aplikacijami, ki so bile podobne naši. Predvsem moramo paziti da sami ne ustvarimo takšnih uporabnikov z aplikacijo, ki ne dela dobro ali ne zadovolji uporabnikovih pričakovanj. Tako je treba pred izdajo aplikacije le to dobro testirati in ugotoviti kakšna pričakovanja imajo uporabniki od aplikacije. Mogoče tudi ni slaba ideja najprej testirati aplikacijo le na delu študentov, na primer samo pri tistih predmetih kjer je osebje najbolj entuziastično glede aplikacije.

2.3.1.4 Zelo aktivni uporabniki

Obstajajo tudi uporabniki, ki bodo aplikacijo uporabljali zelo pogosto in bodo ustvarili veliko vsebin z zelo majhno dodano vrednostjo oz. t.i. nezaželene vsebine. To bi lahko odvrnilo navadne uporabnike od sodelovanja v razpravah, ki jih ponuja aplikacija. Zato je treba poskrbeti z moderiranjem vsebin.

2.3.2 Investicija in vrednost aplikacije za fakultete

Fakultete kot uporabnice oz. stranke zanima predvsem vrednost investicije in koristnost aplikacije za izboljšanje izvajanja pedagoškega procesa. Pri tem lahko kot razvijalci aplikacije vplivamo na oba faktorja.

Pri investiciji so predvsem važni denarni stroški, porabljen čas osebja za integracijo, stroški vzdrževanja in stroški pridobivanja uporabnikov. Pomembno je, da se proizvajalec aplikacije tega zaveda in čim bolje pripravi aplikacijo za te procese.

Za dobro integracijo aplikacije v svoj informacijski sistem potrebujejo znanje. To znanje lahko pridobijo v obliki podpore razvijalcev aplikacije, vendar jim lahko pri tem težavo predstavlja

skrb ob razkrivanju podatkov tretjim osebam. Hkrati jim lahko težavo predstavlja tudi vmešavanje v produkcijsko verzijo informacijskega sistema.

Dobra lastnost fakultet in univerz je to, da ne delujejo z enako intenzivnostjo celotno leto. To omogoči integracijo v času, ko krajši izpadi sistema niso kritični. Tu je treba poudariti, da je fakultetam potrebno predstaviti proces integracije, ki bo ustrezal njihovim časovnim okvirjem. Druga možnost pa je, da se prej integrira rešitev na razvojno verzijo informacijskega sistema. Pomembno je tudi, da začne univerza uporabljati aplikacijo z začetkom šolskega leta. Takrat se vsi študentje in pedagogi seznanijo z učnimi in administrativnimi procesi. To pa zagotavlja boljše pogoje za pridobivanje novih uporabnikov kot to, da se aplikacijo predstavi šele med šolskim letom.

Pri vračilu se je treba opredeliti glede koristi, ki jih taka mobilna aplikacija nudi. Fakultete bodo zanimali pretekli uspešni projekti in študije primerov. Tu je treba določiti parametre, po katerih se bo določala koristnost in aplikacijo ali informacijski sistem prilagoditi tako, da te parametre zajemata. Potrebno je tudi paziti, da so uporabniki mobilne aplikacije pri zbiranju teh podatkov anonimni in da se jih o tem prej obvesti oz. se jim da možnost izbire vklopa te funkcije.

2.4 Konkurenčna prednost razvite aplikacije

Za aplikacijo, ki bo boljša od ali vsaj primerljiva z konkurenco, se je potrebno najprej zavedati prednosti in pomanjkljivosti aplikacij, ki so trenutno na trgu. Potrebno se je tudi opredeliti, kakšne izgube te pomanjkljivosti prinašajo. Glede tega smo se že opredelili v poglavju 2.2 – Analiza funkcionalnosti.

Najbolj očitna stvar, ki jo takoj opazimo pri veliko aplikacijah, je nekonsistentnost uporabniškega vmesnika. Veliko aplikacij vključuje v uporabniški vmesnik kar spletne strani informacijskih sistemov, ki so narejeni za osebne računalnike. To je primer slabe prakse saj prvič: smo odvisni od zunanjih faktorjev in drugič: uporabniški vmesnik je tako slabo dodelan, da se zlahka opazi površnost.

Pri generičnih aplikacijah pride predvsem do izraza bolje izdelan uporabniški vmesnik urnika. Tako vizualno, kot tudi po funkcionalnosti. Pri razvoju naše aplikacije moramo stremeti k podobni kakovosti uporabniškega vmesnika.

Zgornje ugotovitve kažejo na to, da je trg mobilnih aplikacij za pomoč pri organizaciji študija še nerazvit in ni veliko aplikacij, ki bi določale standarde kvalitete. Hkrati je trg razdrobljen na posamezne univerze in tako uporabniki kot odločevalci ne primerjajo aplikacij med seboj, saj so vezani na svoje obstoječe informacijske sisteme in okrnelo ponudbo teh sistemov.

Najpomembnejša konkurenčna prednost moje aplikacije bo dobro izdelan uporabniški vmesnik. Tako s stališča estetike kot same uporabniške izkušnje. Estetsko zasnovana mobilna aplikacija s konsistentnim vmesnikom ustvari močno blagovno znamko in tega na trgu trenutno primanjkuje. Hkrati aplikacija ponuja uporabnikom uporabniško izkušnjo, ki se razlikuje od izkušenj, ki jih ponujajo informacijski sistemi na osebnih računalnikih. Zelo pomembno je, da je uporabniški vmesnik prilagojen tehničnim omejitvam in prednostim, ki jih nudijo mobilne naprave.

Trenutne aplikacije so vse narejene za en specifičen informacijski sistem. Velik del trga študijskih informacijskih sistemov pokrivajo rešitve prej naštetih ponudnikov. Pametno se je vprašati ali se da izdelati različne vtičnike za njihove sisteme, ki bi poenotili komunikacijo med temi sistemi in aplikacijo kot nek vmesni člen.

Razvoj mobilnih aplikacij je zelo zahteven predvsem s stališča zagotavljanja podpore več operacijskim sistemom hkrati. Pri razvoju je treba dejansko paralelno razviti več aplikacij, ki morajo biti konsistentne med seboj. To zahteva veliko truda in usklajevanja. Hkrati lahko različni uporabniški vmesniki zmedejo uporabnike. Potrebno je tudi vložiti več v samo tehnično podporo uporabnikom.

Konkurenčna prednost HTML5 tehnologije v tem, da te tehnologije omogočajo poenoten uporabniški vmesnik na vseh platformah. V našem primeru tako na Android kot iOS operacijskem sistemu. To omogoča hitrejši razvoj, lažjo vzdrževanje in aplikacija je bolj razumljiva za uporabnike. V preteklosti so v zvezi s temi tehnologijami obstajale nekatere omejitve. Predvsem pri operacijskem sistemu Android so se pojavljale težave pri tekočem delovanju aplikacije. Predvsem pri potegih se je uporabniški vmesnik premikal prepočasi, saj Android v osnovi ne uporablja grafičnega procesorja za prikaz vsebin, ki so postavljene v brskalnik. To težavo lahko danes preprečimo z uporabo nestandardnega brskalnika, kot je na primer Crosswalk [46].

2.4.1 Zaščita pred posnemanjem konkurence

Pri programski opremi uveljavljanje patentov v praksi ni ravno mogoče. Tako lahko vsak brez težav naredi podobno aplikacijo in jo prodaja na trgu. Kar lahko kot razvijalci naredimo je, da zaščitimo svojo programsko kodo. Za zagotovitev konkurenčne prednosti lahko razvijemo svoje algoritme in druge elemente, ki jih je težko kopirati. Če si rekel, da bo tvoja prednost GUI a ni tudi designa možno zaščititi v smislu, da preveč podoben od kokurence ne sme biti?

Tukaj naletimo na velik problem saj študijski informacijski sistem Moodle uporablja licenco GPL. Ta licenca je odprtokodna in zahteva izdajo derivativnih del pod to isto licenco. Licenca

sicer ne prepoveduje prodaje programske opreme, mora pa biti zraven zastonj uporabnikom ponujena celotna izvorna koda. V našem primeru to pride v poštev pri potencialni izdelavi Moodle vtičnika za komunikacijo med aplikacijo in informacijskim sistemom Moodle. Morali bi objaviti izvorno kodo vtičnika in s tem bi na primer lahko razkrili pomembne algoritme, ki skrbijo za sinhronizacijo. Ker ima kvaliteten in robusten sistem sinhronizacije veliko dodane vrednosti pri samem projektu, bi ga bilo nespametno odpreti širši javnosti. Za to moramo poiskati drugačno rešitev.

Sinhronizacija med strežnikom in več mobilnimi aplikacijami je zapleten proces, ki lahko ob neoptimalni implementaciji povzroči več različnih vrst težav skrbniku sistema. Od nekonsistentnosti podatkov do velike obremenitve strežnikov. Lahko povzroči tudi nejevoljo končnim uporabnikom saj je lahko aplikacija ob večjih količinah podatkov neodzivna. Še bolj problematičen pa je prenos preko mobilnega omrežja, ki lahko uporabniku povzroči večje stroške.

Rešitev za to je izdelava lastnega »middleware« strežnika, kjer nam ni potrebno objaviti izvorne kode. Hkrati strankam (fakultetam) poenostavimo informacijski proces, saj morajo svoje informacijske sisteme sinhronizirati le z našim strežnikom.

Pri intelektualni lastnini je potrebno opozoriti še na eno stvar. Javascript koda, ki se izvaja v mobilnih aplikacijah, se za razliko od ostalih programskih jezikov ne prevaja v strojni jezik. Zato je nujno to kodo zaščititi. To lahko naredimo z obfuskacijo kode.

2.5 Ključne Ugotovitve analize

V zgornji analizi smo pridobili informacije o naboru aplikacij, ki so že na trgu, o uporabnikih, mobilnih operacijskih sistemih in ponudnikih šolskih informacijskih sistemov. Odločili smo se razviti aplikacijo za FRI, ki bo omogočala komunikacijo med študenti in osebjem. Aplikacija bo v slovenskem jeziku in bo neplačljiva. Pri tem pa se bom osredotočili na dva ključna igralca na trgu mobilnih operacijskih sistemov: to sta Android in iOS, ki sta v aprilu 2015 skupaj pokrivala 84.85% svetovnega trga pametnih telefonov in 94.8% slovenskega. Aplikacija bo vsebovala funkcionalnosti urnika, sporočil oz. novic, indeks, pregled osebja in splošne informacije.

Poglavje 3 Analiza orodij

V tem poglavju bomo pregledali orodja, ki smo jih pred začetkom razvoja analizirali. Opredelili se bomo tudi glede izbranih orodij in pojasnili zakaj smo se tako odločili. Orodja lahko razdelimo na dve podskupini: orodja za samo aplikacijo in orodja za zaledje aplikacije.

3.1 Nabor orodij za aplikacijo

Pri izdelavi aplikacije imamo na razpolago veliko HTML5 orodij, ki nam lahko olajšajo delo. Mnoga pa so potrebna za to, da aplikacija sploh deluje.

3.1.1 Nabor orodij za izdelavo uporabniškega vmesnika

Pri izdelavi uporabniškega vmesnika aplikacije imamo na trgu veliko ponudbo različnih programskih knjižnic. Te se razlikujejo po kompleksnosti, hitrosti oz. zahtevnosti za procesor in spomin, možnostih licenciranja in možnostih spreminjanja in nadgrajevanja.

3.1.1.1 Sencha Touch

Programska knjižnica Sencha Touch [47] je ena od najstarejših HTML5 knjižnic za izdelavo mobilnih aplikacij. Ne nudi samo grafičnih elementov in funkcij za interakcijo, kot so kliki in potegi, ampak tudi podatkovne strukture, funkcije za AJAX komunikacijo, funkcije za obdelavo podatkov in MVC ogrodje. V primerjavi z ostalimi knjižnicami je najbolj obširna in potrebuje največ dela za spreminjanje in nadgradnjo. To pomanjkljivost nadomestijo dobro dodelane metode, ki omogočajo zelo podrobno konfiguracijo. HTML elementi so dinamično generirani znotraj programskega jezika javascript. Hkrati pa se Sencha Touch knjižnica opira na večkratno razširjanje razredov in gradnjo kompleksnejših elementov iz manj kompleksnih delov. To pomeni, da je lahko pod na videz enostavnim HTML elementom mnogo ostalih podelementov. To sicer zelo pripomore k robustnosti vendar pri tem trpi odzivnost aplikacije. Tu se je potrebno opredeliti ali so današnje mobilne naprave zmožne brez opaznih zakasnitev procesirati vse dodatne podatke, ki tukaj nastanejo.

Predvsem je potrebno testirati, kako se aplikacija narejena s Sencha Touch obnaša na operacijskem sistemu Android z nestandardnim brskalnikom Crosswalk, saj ima ta brskalnik veliko večje zmogljivosti glede procesiranja grafičnih nalog. Pri operacijskem sistemu iOS to

ni tako kritično, saj iPadi in iPhoni že v osnovi uporabljajo grafični procesor za prikaz spletnih strani. Enako pa velja tudi za aplikacije, ki so narejene s HTML5 tehnologijami.

Za delo s CSS si knjižnica pomaga s tehnologijo SASS in knjižnico Compass, ki razširja to tehnologijo. To omogoča hitro oblikovanje grafične podobe. Elemente, kot so barvna paleta, oblika gumbov, ikone in ostalo, lahko spreminjamo v glavni konfiguraciji in spremembe se odražajo v celotni aplikaciji. To omogoča hiter razvoj, oteži pa spreminjanje detajlov.

Zraven programskega paketa Sencha Touch je tudi paket orodij Sencha CMD, ki nam omogoča hitro postavljanje projektov in pripravo naše HTML5 programske kode za umestitev v mobilno aplikacijo. Nudi tudi integracijo s programskima paketoma Cordova in Phonegap, ki služita povezavi med tehnologijami HTML5 in operacijskim sistemom mobilne naprave.

Knjižnica Sencha Touch je dostopna pod zastonj komercialno licenco in odprtokodno GPL licenco. To nam omogoča izdelavo aplikacij v komercialne namene, pod pogojem, da ne prodajamo Sencha Touch kot razvojno orodje ali generator mobilnih aplikacij [48].

Zaradi velikega nabora funkcionalnosti, točno določenih smernic razvoja programske opreme in kompleksnosti elementov je potrebno veliko znanja za obvladanje te knjižnice.

3.1.1.2 Ionic Framework

Ionic Framework [49] je novejša knjižnica, ki je v zadnjem času zelo pridobila na popularnosti. Obsega manj funkcionalnosti kot Sencha Touch in se osredotoča predvsem na grafične elemente in zaznavanje uporabniških kretenj. Za ostalo logiko, ki jo moderen uporabniški vmesnik zahteva, se zanaša na zunanje knjižnice kot so AngularJS. Zelo velik pomen je dan majhnosti »DOM«, odzivnosti in preprostosti razvoja. Ta poudarek na preprostost pa pomeni, da v primerjavi s Sencha Touch veliko funkcionalnosti manjka in jih mora razvijalec razviti sam. Čeprav se je veliko lažje naučiti dela s to knjižnico, lahko zaradi pomanjkanja funkcionalnosti razvoj traja enako dolgo ali še dlje.

Tako kot Sencha Touch, tudi knjižnica Ionic Framework nudi orodja za postavitev projekta in povezavo s programskim paketom Cordova.

Ionic Framework je izdana pod MIT licenco in omogoča kakršnokoli komercialno uporabo.

3.1.1.3 React

React [50] je tudi novejša knjižnica, ki nam olajša izdelavo uporabniškega vmesnika. Nastala je pod okriljem Facebooka in Instagrama. Razvijalcem ne prinaša nobenih v naprej izdelanih grafičnih ali interaktivnih elementov. Ta knjižnica je bolj le ogrodje za izdelavo grafičnih

elementov in grajenje kompleksnejših aplikacij z združevanjem le teh. Nudi samo umestitev podatkov v vnaprej definirane maske uporabniškega vmesnika. Ne nudi pa posebnih funkcij za delo s podatki ali komunikacijo z zunanjimi strežniki.

Knjižnica je izdana pod BSD licenco, ki tako kot MIT omogoča kakršnokoli komercialno uporabo.

3.1.1.4 Polymer

Polymer [51] je zbirka HTML elementov, ki so izdelani na podlagi smernic Google Material Design. Polymer s pomočjo javascripta razširjuje HTML sintakso. Tako lahko z zelo malo vložka izdelamo grafične maske za uporabniški vmesnik. Knjižnica nudi tudi nekaj metod za komunikacijo z Google storitvami in metode za komunikacijo preko protokolov kot so bluetooth, https in push sporočila.

Knjižnica je izdana pod licenco tipa BSD in nam omogoča kakršnokoli komercialno uporabo.

3.1.2 Izbor orodja za izdelavo uporabniškega vmesnika

Za izdelavo uporabniškega vmesnika smo se odločili uporabiti knjižnico Sencha Touch. Temu je botrovalo predvsem zelo dobro osebno poznavanje tega programskega paketa. Paket od vseh zgoraj naštetih nudi največ funkcionalnosti in tako lahko najhitreje izdelamo aplikacijo. Ima dolgo zgodovino in je dobro dodelan. Največja ovira je njegova kompleksnost in čas, ki je potreben za učenje uporabe. Ta težava v našem primeru ni bila prisotna saj sem že imel zelo veliko znanja o tej programski knjižnici in sem jo od vseh naštetih najbolje obvladal.

3.1.3 Orodja za integracijo HTML5 aplikacije v gostujoči operacijski sistem

3.1.3.1 Cordova

Programski paket Cordova [52] je ogrodje za umestitev naše HTML5 aplikacije v okolje mobilnega operacijskega sistema. Deluje tako, da se aplikacija zažene v svojem okolju, znotraj te pa se odpre brskalnik. V ta brskalnik se naloži naša HTML5 koda in aplikacija se prikaže na zaslonu. Za komunikacijo z zunanjimi storitvami operacijskega sistema se uporabljajo vtičniki. To so za vsak operacijski sistem posebej napisane metode, ki lahko komunicirajo z metodami v brskalniku.

Na razpolago imamo veliko vtičnikov, ki so podprti s strani projekta, veliko je pa tudi takih, ki so jih razvili zunanji izvajalci. Tako lahko zelo enostavno nadgradimo našo osnovno aplikacijo

z naslednjimi funkcionalnostmi: kamera, odčitovalec QR kode, lokacija, senzor premikanja, push sporočila in še mnogo drugih.

Programski paket Cordova uporablja licenco Apache License 2.0.

3.1.3.2 Phonegap

Phonegap [53] je dejansko identičen programski paket kot Cordova. Preden je nastala Cordova je obstajal samo Phonegap. Nato se je projekt pod svoje okrilje vzelo podjetje Adobe. Podjetje Adobe je nato dalo projekt Phonegap pod okrilje Apache Inkubatorja in nastala je knjižnica Cordova. Zaenkrat se ti dve knjižnici razlikujeta le v imenu. V primeru, da Adobe želi v Phonegap vnesti svoje produkte, lahko to stori brez da vpliva na odprtokodno licenco knjižnice Cordova [54].

Programski paket Phonegap uporablja licenco Apache License 2.0.

3.1.3.3 Crosswalk

Projekt Crosswalk [46] je nastal iz potrebe po enotnem brskalniku predvsem pri operacijskem sistemu Android. Cordova in Phonegap se pri vključitvi brskalnika v aplikacijo zanašata na brskalnik, ki ga zagotovi operacijski sistem. Ti brskalniki pa se zelo razlikujejo glede na verzijo operacijskega sistema kot tudi pri različnih proizvajalcih mobilnih naprav. To pomeni, da se kot razvijalci ne moremo zanašati na to, da bomo imeli vedno na voljo nabor vseh funkcij, ki jih potrebujemo. Hkrati se tudi odzivnosti in na splošno hitrosti lahko močno razlikujejo od naprave do naprave.

Crosswalk rešuje ta problem tako, da namesto standardnega brskalnika v aplikacijo vstavi svojega. Trenutno je največ poudarka na Android brskalniku, možno pa je uporabiti Crosswalk brskalnik tudi pri razvoju za platformo iOS. Pri operacijskem sistemu Android se standarni brskalnik nadomesti kar z brskalnikom, ki ga poganja enako jedro kot brskalnik Chrome za osebne računalnike. Največji doprinos tega je drastično izboljšanje odzivnosti na uporabniške kretnje. To nam tudi omogoči uporabo knjižnice Sencha Touch brez žrtvovanja uporabniške izkušnje.

Crosswalk uporablja licenco tipa BSD.

3.1.4 Orodja za optimizacijo in zaščito izvorne kode

Ker je javascript jezik, ki se ne prevede v strojno kodo, ampak ga brskalnik interpretira, to pomeni, da je ta izvorna koda v naši aplikaciji popolnoma nezaščitena. To lahko predstavlja veliko težavo pri zaščiti intelektualne lastnine. Za rešitev tega se lahko poslužimo orodij, ki

našo kodo naredijo čim manj berljivo. Hkrati pa ta orodja po navadi omogočajo tudi različne vrste optimizacij, za doseganje učinkovitejšega delovanja.

3.1.4.1 Closure Complier

Closure Compiler [55] je orodje, ki ga uporablja Google za optimizacijo svoje javascript kode. Z njim lahko optimiziramo našo programsko kodo tako, da jo očistimo nepotrebne kode in zamenjamo imena spremenljivk s krajšimi. Ta proces v določenem obsegu tudi zmanjša berljivost naše kode.

3.1.4.2 JScrambler

JScrambler [56] je storitev, ki zraven funkcionalnost za optimizacijo, nudi tudi vrsto drugih funkcionalnosti za zaščito izvorne kode. Poleg obfuskacije nudi tudi zaklep na domeno, zaklep na določen brskalnik ali operacijski sistem, zaščito pred orodji za razhroščevanje in zaklep ob določenem času.

Storitev ni brezplačna in je na voljo v različnih paketih, ki se cenovno gibljejo med 372€ na leto do 1020€ na leto [4]. Storitev deluje tako, da se izvorna koda prenese na njihove strežnike, tam se obdela, nato pa se prenese nazaj. Na voljo je tudi »enterprise« paket, ki omogoči izvajanje algoritmov zaščite in optimizacije na lastnih strežnikih.

3.1.5 Orodja za gradnjo aplikacije

Čeprav nam poenoten razvoj omogoča hitro izdelavo HTML5 aplikacije, je pri umestitvi naše HTML5 kode potrebno izvesti kompleksnejše postopke. Ker razvoj hibridnih aplikacij ni standardna oblika razvoja, se tudi ne moremo vedno zanašati na standardna orodja.

Za razvoj aplikacije nismo uporabljali posebnih IDE orodji, ampak smo se zanašali na navaden urejevalnik besedila. Vsa ostala orodja, ki smo jih potrebovali pa so prišla zraven knjižnic Sencha Touch in Cordova. Programe za gradnjo aplikacije smo poganjali preko terminalnega okna. Na prvi pogled se zdi to oteževanje procesa izgradnje aplikacij, vendar nam to omogoča avtomatizirano izgradnjo na strežnikih.

Izgradnja same aplikacije je potekala v naslednjih korakih:

1. Predelava HTML5 kode v produkcijsko verzijo, kjer so se vse datoteke združile v eno
2. Optimizacija in obfuskacija produkcijske kode
3. Umestitev produkcijske kode v ogrodje Cordova
4. Izdelava aplikacije tipa »development« za Android
5. Izdelava aplikacije tipa »production« za Android in podpisovanje aplikacije s certifikatom
6. Izdelava aplikacije tipa »development« za iOS
7. Izdelava aplikacije tipa »production« za iOS in podpisovanje s certifikatom

3.1.5.1 Proces izdelave aplikacije za Android

Pri gradnji aplikacije za Android smo za to uporabili orodja iz programskega paketa Cordova. Nato je bilo treba aplikacijo samo še podpisati s programom jarsigner.

3.1.5.2 Proces izdelave aplikacije za iOS

Pri gradnji aplikacije je bil proces bolj zapleten saj orodja iz programskega paketa Cordova niso zadostovala. Zato smo aplikacijo zgradili s pomočjo podprogramov razvojnega paketa XCode.

Razvoj aplikacije za operacijski sistem iOS zahteva v zadnji fazi razvoja izdelavo aplikacije na Apple računalniku. Za končno aplikacijo potrebujemo orodje XCode. Ker je XCode Applova licenčna programska oprema in deluje samo v Applovem operacijskem sistemu OSX, drugo orodje, s katerim bi lahko zgradili aplikacijo na drugih operacijskih sistemih ne obstaja.

Razvit je bil avtomatičen proces, kjer se produkcijska koda presname na OSX strežnik in tam se generira iOS aplikacija s pomočjo razvojnega paketa XCode. Nato se aplikacija prenese nazaj.

3.2 Orodja za zaledje (backend) aplikacije

Za izdelavo strežnika za komunikacijo z aplikacijo smo se odločili, da postavimo celotno infrastrukturo v oblaku. Tej odločitvi so botrovali poceni zagonski stroški in preprosta možnost razširjevanja kapacitet.

3.2.1 Storitve v oblaku

Na trgu obstaja kar nekaj ponudnikov storitev računalništva v oblaku. Največji igralci so:

- Amazon Web Services
- Google Cloud Platform
- Heroku
- Rackspace

Zaradi največjega nabora funkcionalnosti in preteklih izkušenj s storitvijo, sem za razvoj aplikacije izbral Amazon Web Services.

Amazon Web Services ali krajše AWS je skupina storitev, ki jo nudi podjetje Amazon. Zaradi velikega števila teh storitev, se bomo seznanili samo s tistimi, ki so pomembne za naš uporabniški primer.

3.2.1.1 EC2 (procesiranje)

Storitev EC2 nam omogoča najem virtualnih v Amazonovih podatkovnih centrih. Lahko izberemo lokacijo teh strežnikov in velikost. Na voljo so tudi različni tipi strežnikov, ki so optimizirani za svoje naloge. Tako lahko izbiramo med strežniki, ki so optimizirani za: računske operacije, spominske operacije, grafične operacije in operacije z datotekami.

Na strežnike lahko naložimo različne Linux in Windows operacijske sisteme. Operacijski sistem OSX ni podprt, saj Apple ne dovoli uporabe svoje programske opreme na strojni opremi drugih proizvajalcev. Datoteke, na katerih so shranjeni operacijski sistemi, imenujemo namestitvena slika ali Amazon Machine Image (AMI). Za enostavno kopiranje našega operacijskega sistema na druge virtualne strežnike, lahko ustvarimo tako sliko iz operacijskega sistema našega trenutnega strežnika. Prenesli se bodo tudi vsi programi, ki smo jih izdelali ali namestili. V AWS trgovini (https://aws.amazon.com/marketplace), lahko dobimo tako veliko programov zunanjih proizvajalcev. Ti so lahko brezplačni ali pa moramo za njih plačevati naročnino.

Ob kreiranju novega strežnika temu nastavimo varnostno skupino. Vsaka varnostna skupina ima svoj privatni ključ za dostop preko protokola SSH. V tem procesu tudi nastavimo, katera vrata (porti) naj bodo odprta zunanjemu svetu.

Amazon za to storitev nudi eno leto brezplačno preizkusnega obdobja na najmanjšem možnem strežniku. Sicer pa se beleži, koliko časa je strežnik aktiven in se obračuna vsako delovno uro. Strežnike lahko kadarkoli ugasnemo in jih nato po potrebi spet prižgemo.

Vsakemu strežniku je ob zagonu dodeljen IP naslov. Ta naslov se ob vsakem ponovnem zagonu zamenja. Če želimo to preprečiti, potem lahko zakupimo tako imenovani elastični IP. Ta IP se ohrani, tudi če je strežnik ugasnjen. To je zelo dobrodošlo predvsem pri strežnikih, ki so dostopni javnosti preko domene. Potrebno pa je to storitev plačati.

3.2.1.2 Lambda (procesiranje)

AWS Lambda je tehnologija, ki so jo predstavili šele pred kratkim. Omogoča izvedbo programske kode brez strežnika in operacijskega sistema. Svojo kodo lahko preprosto nalažimo na AWS Lambda strežnike in določimo, ob katerih dogodkih naj se izvede. Ti dogodki so lahko: prenost datoteke v storitev S3, push sporočilo, prihod podatkov v AWS Kinesis in šele pred kratkim dodana možnost, AWS API gateway zahteva. Ta storitev nam omogoča izvajanje funkcionalnosti brez strežnika, kar je dobro pri zagotavljanju resursov. Na klasičnem strežniku namreč lahko neka metoda zavzema preveč strežniških kapacitet in s tem zavira ostale. Na AWS Lambda bi te metode delovale neovirano.

Storitev se obračuna tako, da se beleži čas in spominski prostora, ki ga je funkcija porabila.

3.2.1.3 S3 (hranjenje in dostava datotek)

Datoteke lahko shranjujemo pri AWS na več načinov. Najbolj uporabljana storitev za to pa je AWS S3. S3 nam omogoča shranjevanje poljubno število datotek poljubno velikih datotek. Datoteke so znotraj S3 zaščitene z enkripcijo.

Tako kot pri datotečnih sistemih informacijskih sistemov, lahko tudi tukaj določimo skupine uporabnikov in jim dodelimo dovoljenja, kot so dovoljenja za branje in pisanje. Lahko tudi odločimo ali so datoteke javne ali zasebne. Če datoteko označimo za javno, potem je dostopna preko URL-ja.

3.2.1.4 CloudFront (hranjenje in dostava datotek)

CloudFront je storitev distribucije vsebin ali content delivery network (CDN). Storitev lahko uporabljamo skupaj z S3 ali pa hranimo vsebine na svojem strežniku. CloudFront skrbi za to, da so statične vsebine vedno na voljo uporabnikom in se za dostavo uporabi najbolj optimalna pot. Tako so vsebine razpršene po Amazonovih podatkovnih centrih po vsem svetu. Ko uporabnik zahteva vsebine, CloudFront poskrbi, da mu jih postreže najbližji podatkovni center.

3.2.1.5 DynamoDB (podatkovne baze)

DynamoDB je Amazonova implementacija nerelacijske baze podatkov ali NoSQL. Omogoča hranjenje podatkov v obliki JSON, kjer je lahko vsak vnos velik največ 400 KB. Ker ima ta tip podatkovnih struktur veliko manjši nabor funkcionalnosti kot ga imajo relacijske baze, je potrebno v večini primerov izvesti operacije nad podatki v sami aplikaciji. To pomeni, da mora razvijalec sam razviti funkcionalnosti za preverjanje podatkov ob vnosu. Tu je potrebno biti posebej pazljiv, da uporabniki ne vnesejo podatkov, ki ne ustrezajo podatkovnemu modelu.

Prednost nerelacijske baze nad relacijsko je predvsem v ceni in neomejenih možnosti razširjanja. Pri relacijskih bazah se največkrat izvajajo zahtevne iskalne operacije kar v sami podatkovni bazi. Te operacije so po navadi optimizirane z indeksiranjem, kar zagotavlja hitre poizvedbe nad veliko množico podatkov.

Vseeno pa to v veliko primerih ni dovolj in slej ko prej zahteve uporabnikov presežejo zmogljivosti baze. Ker so te baze vezane na točno določene strežnike in je sinhronizacija med več relacijskimi bazami zelo zahtevna, je zelo težko take sisteme nadgrajevati. Nerelacijske baze pa so realizirane prav z namenom, da ti dve težavi ne obstajata. Podatki so lahko razpršeni v oblaku in ni nujno, da so vsi na istem fizičnem strežniku.

Do njih lahko dostopamo s ključem oziroma več ključi. Pri DynamoDB imamo dva tipa ključev, ki ju lahko uporabimo za poizvedbo: »hash key« in »range key«. Poizvedba s »hash« ključem nam vrne vse objekte, ki vsebujejo ta ključ – lahko imamo več takih objektov. Poizvedba z »range« ključem nam pa omogoča iskanje vseh objektov, ki so med dvema vrednostima. Na primer med dvema datumoma. Lahko uporabimo dva tipa poizvedb: »query« in »scan«. »Query« poizvedba potrebuje »hash« ključ in nam omogoča hitre in poceni operacije samo nad podatki, ki vsebujejo ta »hash« ključ. Operacija »scan« pa izvede poizvedbo nad celotno tabelo.

Zelo je pomembno, da se operacij »scan« izogibamo, saj se uporaba DynamoDB obračunava glede na število branj in pisanj. S tega stališča so operacije »query« pri velikih tabelah še toliko bolj dobrodošle.

Lahko bi trdili, da pri nerelacijskih bazah vse te okrnitve ne odtehtajo nižje cene in enostavnega razširjevanja. V končni fazi, če potrebujemo zapleteno operacijo nad velikim številom podatkov, potem bomo morali nekje porabiti računalniške resurse: procesorski čas in spominski prostor, neglede na to ali se to zgodi v bazi podatkov ali na nekem drugem strežniku. V tej enačbi pa je potrebno upoštevati še eno spremenljivko. To je, da se dandanes večina algoritmov izvede kar na mobilnih napravah. Uporabniki ne pričakujejo, da bodo mobilne aplikacije delovale kot spletne strani, kjer se za vsako novo poizvedbo čaka, da se podatki vrnejo iz oddaljenega strežnika. Za zagotavljanje odzivnosti potrebujemo podatke na voljo takoj in

čakanje na te podatke na primer pol sekunde v praksi ne zadostuje. Tako je najbolje, da povezava med aplikacijo in glavnim strežnikom in aplikacijo, skrbi le za sinhronizacijo podatkov. Sami algoritmi pa naj se izvajajo kar na aplikaciji sami. To tudi zelo zmanjša strežniške zahteve in stroške.

3.2.1.6 Route53 (omrežje)

Route53 je storitev domenskega strežnika in domenski register. S to storitvijo lahko registriramo domeno in upravljamo njene DNS nastavitve. Deluje enako kot vsi ostali domenski registri.

3.2.1.7 Cognito (mobilne storitve)

Storitev AWS Cognito nam omogoča shranjevanja stanja mobilne aplikacije uporabnika in sinhronizacijo tega stanja z drugimi napravami, na katerih mogoče uporablja isto aplikacijo. Ta storitev nam omogoča boljšo uporabniško izkušnjo, saj je stanje aplikacije na vseh napravah enako.

3.2.1.8 Device Farm (mobilne storitve)

AWS Device Farm je novejša Amazonova storitev, ki nam omogoča testiranje naše aplikacije na velikem številu mobilnih naprav. Storitev podpira Android in iOS aplikacije. To omogoči razvijalcem zelo kvalitetno testiranje brez investicije v dejanske fizične naprave.

3.2.1.9 Mobile Analytics (mobilne storitve)

Mobile Analytics nam omogoča zbiranje in analizo podatkov o uporabi naše aplikacije. V svojo aplikacijo lahko naložimo Mobile Analytics programski paket, kar nam omogoča, da beležimo konkretne uporabniške primere in ne samo statistike kot so število uporabnikov in čas seje. Lahko določimo korake in beležimo ali so uporabniki uspešno opravili te korake. To nam da boljši vpogled v kvaliteto uporabniškega vmesnika.

3.2.1.10 SNS (mobilne storitve)

Velikokrat je v mobilnih aplikacijah najbolj dobrodošla funkcionalnost »push« sporočil. Ta nam omogoča pošiljanje obvestil uporabnikom na našo pobudo in ne samo takrat, ko jih uporabnik zahteva. Na voljo so nam protokoli za vse možne mobilne operacijske sisteme.

3.2.1.11 API gateway (aplikacijske storitve)

API gateway je novejša storitev na AWS. Na prvi pogled ni nič posebnega in bi jo lahko enostavno realizirali na EC2 infrastrukturi. Prav to smo morali delati do sedaj. API ni nič drugega kot strežnik brez uporabniškega vmesnika. Kar je pomembno pri tej AWS storitvi pa

je, da omogoča povezavo s storitvijo AWS Lambda. To prej ni bilo mogoče in se ni dalo izvajati Lambda funkcij kot odgovor na navadne HTTP/HTTPS poizvedbe. Zdaj je to mogoče.

3.3 Programska oprema za zaledje aplikacije

3.3.1 Node.js

Node.js [57] je programsko okolje, ki nam omogoča izvajanje javascript programske kode brez brskalnika. Uporablja se predvsem kot programska oprema za spletne strežnike. Omogoča asinhrono delovanje kar pomeni, da uporabniške zahteve ne čakajo ena na drugo in zahtevnejši procesi ne zavirajo drugih manj zahtevnih. Javascript koda se interpretira s pomočjo jedra V8 (https://developers.google.com/v8/), ki ga uporablja tudi brskalnik Google Chrome.

Skupaj z Node.js se uporablja program za namestitev dodatnih modulov »npm« ali »node package manager«. S tem programom lahko namestimo mnogo odprtokodnih dodatkov. Izmed njih tudi vse spodaj naštete.

3.3.2 Express

Express [58] je programska knjižnica, ki deluje kot ogrodje za spletni strežnik. Omogoča nam preprosto rešitev za odzivanje na uporabniške HTTP zahteve kot so GET in POST. Dodamo pa lahko tudi veliko drugih funkcionalnosti, kot so knjižnice za delo s sejami, piškotki in ostale funkcionalnosti, ki jih ponujajo spletne tehnologije.

3.3.3 Vogels

Vogels [59] je adapter, ki nam omogoča preprosto uporabo DynamoDB API-ja v node.js okolju. Osnovna komunikacija z podatkovno zbirko v oblaku DynamoDb poteka preko HTTP protokola. V vsakem programskem jeziku moramo nato sami sprogramirati te klice. Na voljo so nam tako adapterji, v katerih so vse take funkcije že napisane in jih moramo samo vključiti v naš projekt. Amazon Web Services pri DynamoDB storitvi v sklopu svojih paketov za razvoj podpirajo tri programske jezike: Javo, .NET in PHP. Ker programskega jezika javascript ni med njimi smo se poslužili neuradnega adapterja Vogels.

3.3.4 node-imagemagick-native

Program node-imagemagick-native [60] je adapter za programsko knjižnico ImageMagick. Ta knjižnica nam, omogoča manipulacijo slik. Na voljo je veliko takšnih adapterjev vendar vsi ostali komunicirajo z ImageMagick preko programskega terminala. Le ta adapter nam

omogoča, da komuniciramo s knjižnico direktno preko programa. To nam omogoči hitrejše delovanje in večjo možnost dopolnjevanja funkcionalnosti.

3.3.5 ImageMagick

ImageMagick je programska knjižnica za delo s slikami. Obstaja veliko verzij te knjižnice pod različnimi imeni saj je napisana v mnogo programskih jezikih. Mi bomo uporabljali C++ verzijo Magick++ [61] saj je ta verzija kompatibilna z našim adapterjem.

3.4 Izbor orodij za zaledje (backend) aplikacije

Za zaledje naše aplikacije smo potrebovali storitve:

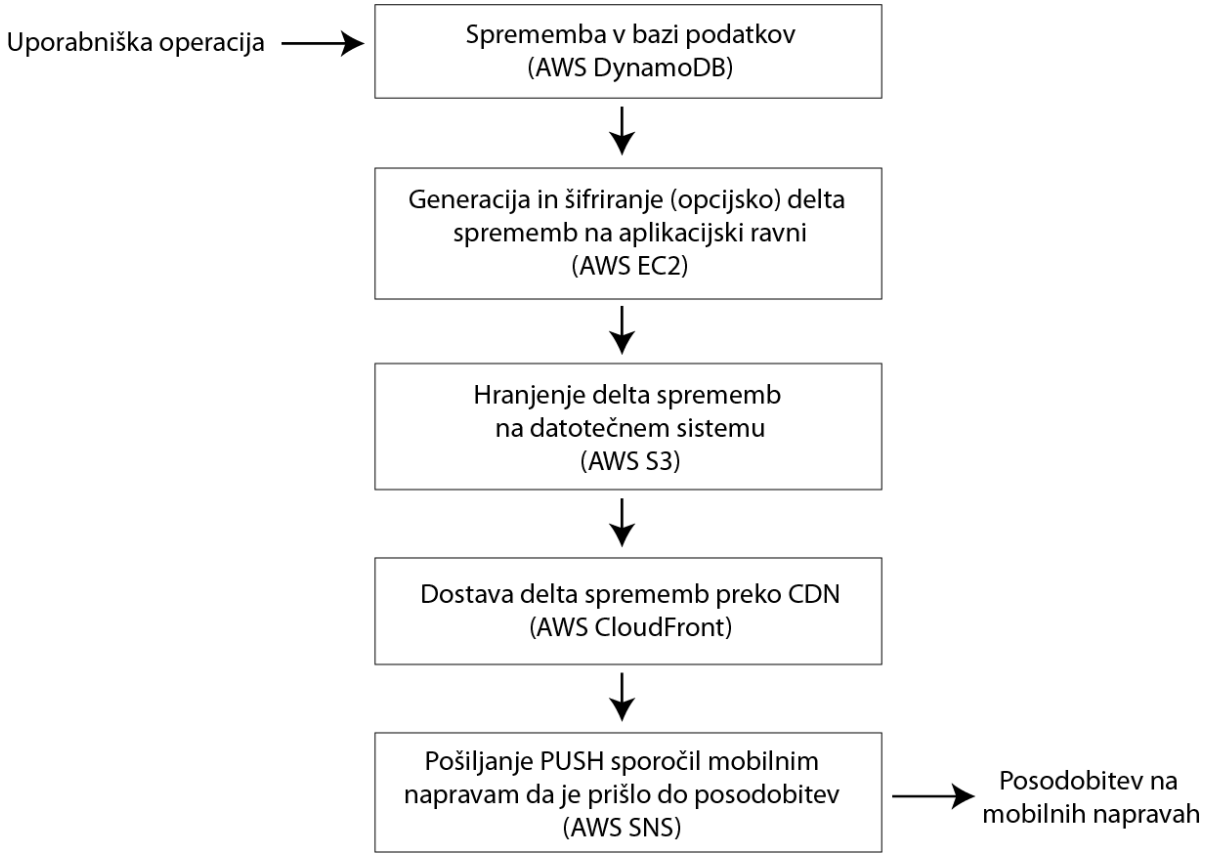
- EC2 za postavitev spletnega strežnika, ki ga poganja programska oprema node.js. Na tem strežniku gostuje administrativni vmesnik za urejanje vsebin, ki so prikazane v mobilni aplikaciji. Hkrati se na njem izvajajo vse operacije za pripravo podatkov za sinhronizacijo. Generirajo se vse potrebne slike s pomočjo programske opreme ImageMagick in vzpostavljena je komunikacija med strežnikom, podatkovno bazo DynamoDB in sistemom za upravljanje z datotekami S3.
- DynamoDB se uporablja za hranjenje vseh podatkov, ki so potrebni za aplikacijo.
- S3 se uporablja za hranjenje datotek za sinhronizacijo med mobilno aplikacijo in strežnikom. Hranijo se tudi druge datoteke kot so slike.

Strežniško programsko opremo smo razvili na EC2 instanci, ki jo poganja operacijski sistem Linux. Strežniško programsko opremo smo razvili na platformi node.js s povezavo modulov: Express, Vogels, node-imagemagick-native in imageMagick.

Poglavje 4 Razvoj aplikacije

V tem poglavju bomo opisali, kako smo dosegli zadnji cilj diplomske naloge, to je izdelava mobilne aplikacije, za pomoč pri organizaciji študija za FRI. Pri razvoju aplikacije in njenega zaledja smo se najprej osredotočili na načrtovanje. Ugotoviti je potrebno: kakšne so strojne in programske zahteve za aplikacijo, zasnovati podatkovni model, kako bo potekala sinhronizacija med informacijskim sistemom in aplikacijo in kako bo poskrbljeno za varnost in integriteto komunikacije. Pri tem smo poskušali vpeljati čim več že razvitih in preverjenih rešitev, ki so vključene v izbrane tehnologije.

Celoten sistem mora implementirati naslednji proces obdelave podatkov:



Slika 4.1: Proces priprave datotek za sinhronizacijo pri spremembi v podatkovni bazi

4.1 Zasnova podatkovnega modela

Podatkovni model moramo oblikovati tako, da ustreza 2 kriterijema:

- V zaledju morajo biti podatki strukturirani tako, da jih lahko hranimo v nerelacijski bazi podatkov v oblaku. V našem primeru je to DynamoDB.
- Podatki morajo biti strukturirani tako, da omogočajo hitro, podatkovno optimizirano in varno sinhronizacijo med zaledjem in aplikacijo.

Podatkovne strukture v obeh sistemih, Sencha Touch na strani mobilne aplikacije in podatkovna baza v zaledju, so relativno preproste. Ne omogočajo kompleksnejših operacij, kot nam jih nudijo relacijske baze. O samem poteku sinhronizacije bomo govorili v naslednje poglavju. Pri podatkovnem modelu pa je potrebno povedati, da mora omogočati beleženje sprememb pri vnosih v podatkovno bazo za potrebe optimizacije sinhronizacije.

4.2 Sinhronizacija med aplikacijo in strežnikom

Sinhronizacija mora biti implementirana tako, da se, kadar se le da, prenašajo samo spremembe, ki so nastale po zadnji povezavi med mobilno aplikacijo in zaledjem. Treba se je opredeliti, kateri podatki v aplikaciji so javne in kateri zasebne narave. Javne podatke lahko hranimo na S3 strežniku in so lahko dostopni vsakomur. S tem prihranimo na strežniških stroških saj ne potrebujemo EC2 strežnika, ki je ena izmed najdražjih komponent v AWS ponudbi storitev v oblaku.

Zasebne podatke je potrebno zagotoviti samo uporabnikom, ki so prijavljeni v aplikacijo. Za to potrebujemo EC2 strežnik, saj ostale AWS storitve ne omogočajo dela s sejami.

Za zagotavljanje integritete podatkov je poskrbljeno z »MD5 hashiranjem« celotne zbirke podatkov na obeh straneh.

Tak način predstavlja za strežnik veliko dodatnega dela. Strežnik mora ob vsakem vnosu primerjati potencialno zelo velike zbirke podatkov in pripraviti nove datoteke, ki se potem prenesejo na S3 strežnike. Ker pa je sprememb v bazi podatkov v primerjavi s poizvedbami iz mobilnih naprav zelo malo, se taka rešitev splača. Popolnoma odpravimo tudi potrebo po širjenju infrastrukture z naraščanjem števila uporabnikov saj storitvi S3 in CloudFront ne potrebujeta vnaprej zakupljenih kapacitet in jih plačujemo popolnoma po potrebi.

4.2.1 Sinhronizacija podatkov

Sinhronizacija podatkov poteka po naslednjem postopku:

1. Mobilna aplikacija dobi PUSH obvestilo o tem, da je nastala posodobitev na strežniku in v kateri zbirki podatkov je nastala.

2. Mobilna aplikacija primerja svojo verzijo zbirke podatkov z verzijo, ki je na strežniku in ugotovi, koliko in katere delta posodobitve mora prenesti.

3. Na mobilno aplikacijo se prenesejo potrebne posodobitve.

4. Naredi se MD5 hash čez celotno zbirko podatkov na napravi in se primerja z MD5 hashem iz strežnika.

5. V primeru, da se hasha ne ujemata, se na mobilno napravo še enkrat prenese celotna zbirka podatkov.

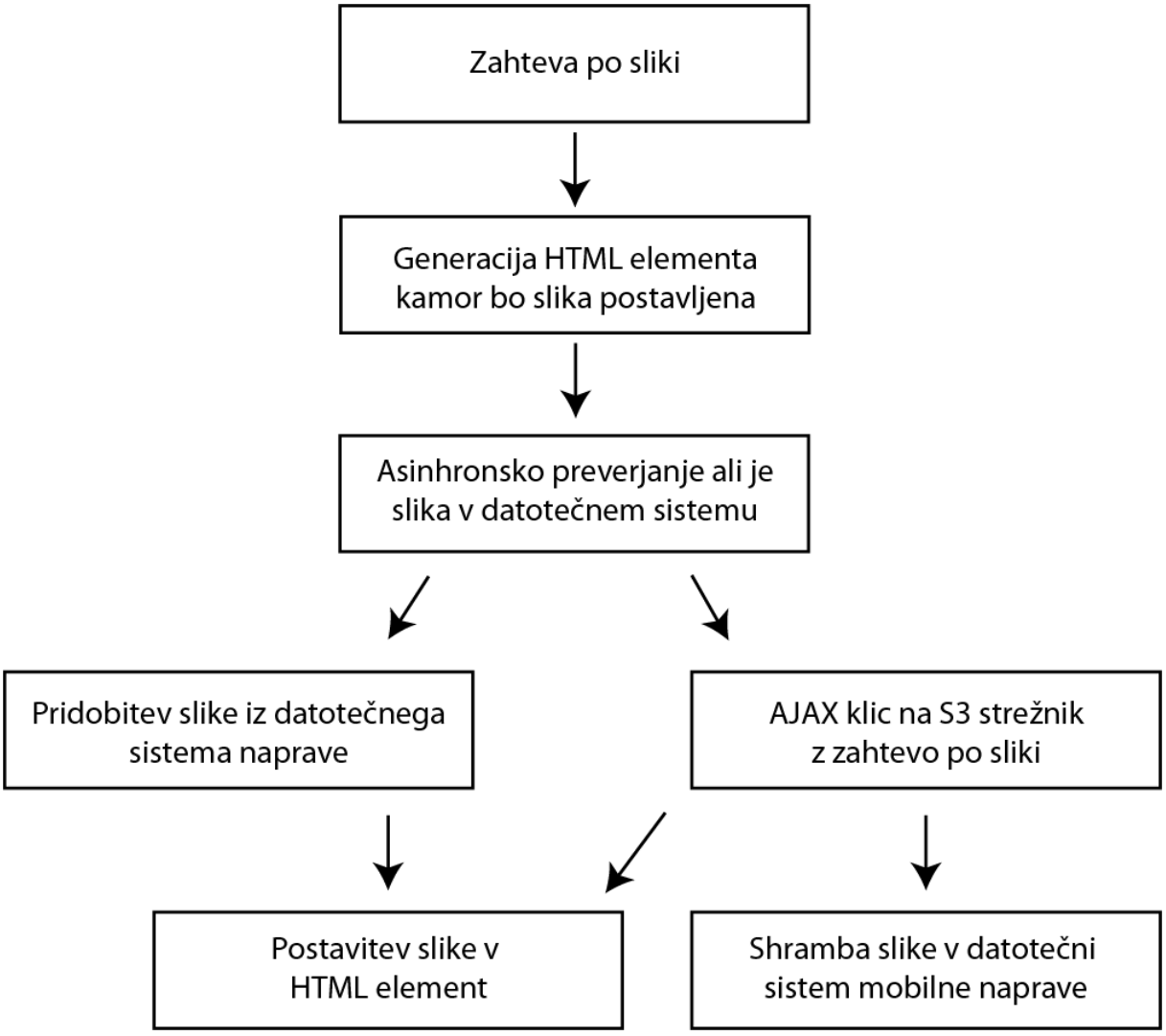
S tem postopkom razbremenimo zaledno infrastrukturo. Celotni podatki, delta spremembe, MD5 hash zbirke in številka verzije generiramo samo ob spremembi podatkov na strežniku. Te spremembe hranimo na datotečnem sistemu AWS S3. Uporabniki dobijo te podatke na svoje mobilne naprave preko sistema za dostavo vsebin (CDN) in sploh ne pošljejo poizvedbe na strežnik. S tem postopkom lahko v celoti izkoristimo vse tehnologije v oblaku, ki so nam na voljo.

4.2.2 Sinhronizacija datotek

Pri hranjenju podatkov v DynamoDB tabelah omejeni na 400KB. To je za hranjenje datotek, kot so slike, premalo in se ne moramo zanašati na to, da nikoli ne bomo presegli take količine podatkov. Slike in ostale datoteke tako hranimo v datotečnem sistemu S3, v DynamoDB tabeli »resources« pa hranimo URL povezavo do teh datotek.

Datoteke se ne sinhronizirajo tako kot navadni podatki, saj so količine podatkov veliko večje. Na primer: pri seznamu osebja ima vsaka oseba svojo sliko. Za vsako osebo so tako na strežniku generirane slike (thumbnaili), ki so na voljo v različnih velikostih in oblikah. Če ima vsaka oseba 10 različnih slik in je osebja okoli dvesto, bi to pomenilo 2000 datotek, ki so v povprečju velike 10KB. To nanese 20MB na sinhronizacijo. Datoteke se zato sinhronizirajo samo takrat, ko jih potrebujemo. V našem primeru je to takrat, ko se mora slika prikazati na ekranu. V

aplikaciji moramo tako implementirati sistem, ki omogoča nalaganje datotek iz strežnika s pomočjo AJAX povezave, hkrati pa dopušča možnost nalaganja datotek iz datotečnega sistema mobilne naprave. Spodnji diagram prikazuje proces pridobivanja in prikaza slike:



Slika 4.2: Sinhronizacije in prikaza slike v aplikaciji

4.3 Obdelava slik

Današnji brskalniki veliko bolje podpirajo prikaz slik v drugačnih velikostih, kot je originalna datoteka. Danes je možno implementirati spremembe velikosti in spremembo oblike slike kar v brskalniku mobilne naprave. Tak primer je na primer generacija thumbnailov.

Takšna implementacija zahteva veliko manj dela, kot generacija slik na strežniku. Kljub temu pa so slabosti te implementacije pri performansih aplikacije preveč moteče, da bi jo lahko uporabili. Te slabosti so:

- Sinhronizirati bi morali večjo datoteko, kar bi zahtevalo več časa in slabšo odzivnost aplikacije
- Za prikaz je potrebno več procesorskega dela kar še posebej poslabša odzivnost ob potegih in ostali človeški interakciji
- Vseeno je treba na strežniški strani postaviti omejitve glede največjih dimenzij in velikosti slike, hkrati pa tudi pretvoriti prevelike slike v prave dimenzije.

Druga možnost je implementacija sistema pretvorbe slik na strežniku. Ta možnost zahteva veliko več truda, vendar odpravi vse pomanjkljivosti prejšnjega sistema. Tako smo se odločili za to možnost. Rešitev je bila realizirana s pomočjo NodeJS vtičnika node-imagemagick-native. Vtičnik omogoča:

- Pretvorbo med formati slik
- Megljenje slike (blur)
- Povečanje ali pomanjšanje slike
- Nagib in obračanje slike
- Združevanje več slik v kompozicijo
- Pridobitev informacije o sliki in posameznih točkah

Vtičnik sicer omogoča le malo funkcionalnosti, ki jih drugače nudi knjižnica za delo s slikami ImageMagick. Tako ne omogoča uporabe ene od slik kot maske pri kompoziciji. To pomeni, da postavimo črno-belo sliko pred sliko z vsebino in tam , kjer je črna barva, postane druga slika transparentna, tam, kjer je bela barva pa ostane takšna kot je. To funkcionalnost smo

potrebovali za generacijo profilnih slik in smo jo implementirali s spremembo funkcije za kompozicijo.

4.4 Izdelava »po meri« modulov za aplikacijo

Vsi elementi, ki smo jih potrebovali za gradnjo naše aplikacije, niso bili na voljo v knjižnici Sencha Touch in smo jih morali narediti sami. Za potrebe naše aplikacije smo knjižnico Sencha Touch razširili z naslednjimi elementi:

- Razred Ext.ux.EnterpriseDBStore, ki omogoča sinhronizacijo podatkov s strežnikom
- Razred FRI.ux.TouchCalendarView, ki prikazuje koledar
- Razred FRI.ux.MaterialButton, ki ga uporabljamo namesto Sencha Touch gumba Ext.Button in implementira Waves gumb (http://fian.my.id/Waves/)
- Razred FRI.ux.MaterialCheckbox, ki ga uporabljamo namesto Sencha Touch elementa Ext.field.Checkbox in podpira animacijo
- Razred FRI.ux.StaffSearchButton, ki ga uporabljamo namesto Sencha Touch elementa Ext.field.Text in najprej deluje kot gumb, ob pritisku pa se razširi v polje za vnos teksta

4.4.1 Ext.ux.EnterpriseDBStore

Razred Ext.ux.EnterpriseDBStore je razširjen standarden Sencha Touch razred Ext.data.Store. Ext.data.Store razred uporabljamo kot podatkovni model v MVC načinu razvoja naše aplikacije. Omogoča nam hranjenje podatkov za čas, ko je aplikacija prižgana, in funkcije iskanja, sortiranja in združevanja. Ta razred pa ne omogoča hranjenja podatkov na napravi v času, ko aplikacija ne deluje. To pa je potrebno za sinhronizacijo v naši aplikaciji.

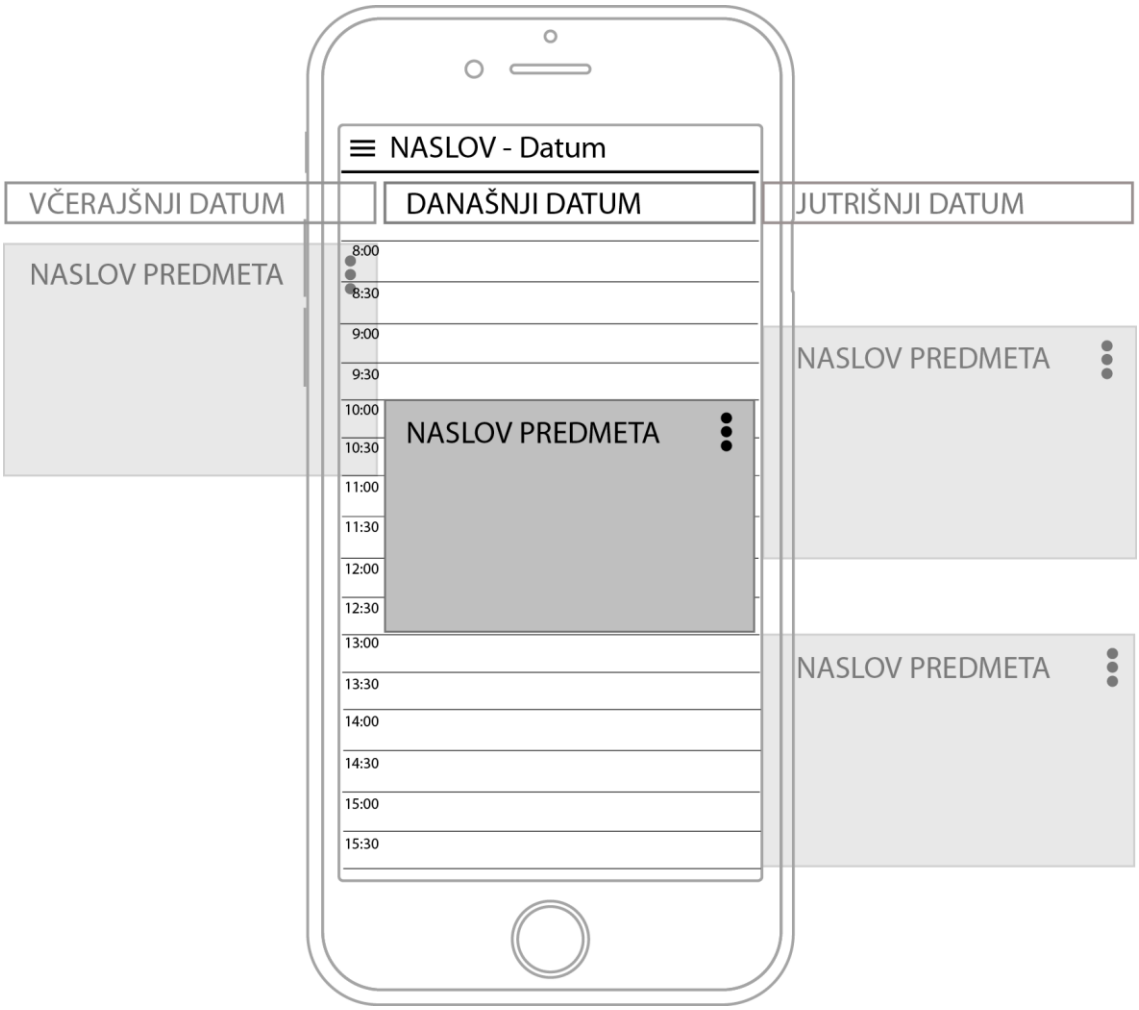
Zaradi tega smo ustvarili nov razred, ki to podpira. Ext.ux.EnterpriseDBStore uporablja programsko knižnjico Lawnchair, ki deluje kot adapter za različne persistentne načine shranjevanja podatkov: blackberry-persistent-store, dom, gears-sqlite, ie-userdata, webkit-sqlite, indexed-db, memory. Za našo aplikacijo smo izbrali webkit-sqlite.

Hkrati pa so bili implementirani tudi algoritmi delta posodabljanja, ki v ozadju opravljajo vse operacije sinhronizacije in zagotavljajo integriteto podatkov. Delta posodabljanje deluje na podoben princip, kot pri »VCS« (version control system) sistemu git [62]. Ob spremembi podatkov se na strežniku ustvari datoteka, ki vsebuje spremembe med predzadnjo in zadnjo verzijo podatkov. Mobilna aplikacija nato pridobi vse datoteke, ki so potrebne za to, da se

verzija na mobilni napravi izenači z verzijo na strežniku. Za posodabljanje glavne zbirke podatkov s podatki iz delta datotek smo uporabili knjižnico »jsondiffpatch« [63]. Jsondiffpatch knjižnica uporablja dve metodi: diff in patch. Metoda diff iz dveh zbirk podatkov določi razliko. Metoda patch pa iz podatkov in razlike sestavi nove podatke. Metodo diff uporabljamo na strežniški strani, da določimo delta objekte. Metodo patch pa uporabljamo v mobilni aplikaciji, da združimo stare podatke z novimi delta objekti.

4.4.2 FRI.ux.TouchCalendar

FRI.ux.TouchCalendar je razred, ki skrbi za prikaz koledarja oz. urnika in uporabniško interakcijo. Zaradi kompleksnosti in računske zahtevnosti, izgradnja tega vizualnega elementa ni bila mogoča samo z uporabo standardnih Sencha Touch elementov. Na spodnji sliki je v grobem prikazana struktura tega urnika:



Slika 4.3: Struktura urnika

Za uporabniku na videz preprosto uporabniško izkušnjo je bilo potrebno izdelati kompleksno logiko in veliko optimizacij pri prikazu grafičnih elementov.

Najprej se mora koledar prilagajati velikosti zaslona. Prilagajata se število kolon in širina kolone. Vsaka kolona predstavlja en dan. Na pametnem telefonu je po navadi v stoječem položaju prikazana le ena kolona. V ležečem položaju pa so lahko prikazane dve ali tri. Pri tabličnih računalnikih pa je lahko na zaslonu prikazanih do pet kolon hkrati. Število kolon se mora nemudoma prilagoditi ob spremembi položaja naprave in to na tak način, da uporabnika ne zmede in mu ne oteži uporabniške izkušnje.

Koledar mora tudi omogočati interakcijo s potegom v obeh smereh. Tako vertikalno kot tudi horizontalno. Vertikalni poteg premakne predstavlja premik na urni časovnici, horizontalni premik pa predstavlja spremembo dneva. Potega se ne smeta med sabo mešati. Uporabnik nikoli ne naredi kretnje popolnoma vertikalno ali popolnoma horizontalno. Tu mora aplikacija dobro registrirati, za kateri poteg gre in drugo možnost za čas potega izključiti.

Glava aplikacije vsebuje v naslovu prvi in zadnji datum, ki sta prikazana na kolonah koledarja. Tudi ta naslov se mora spreminjati skupaj s horizontalnimi potegi.

Na levi strani so označene ure in minute. Da so vidne, je potrebno kolone, ki predstavljajo dni, zamakniti za določeno razdaljo v desno. Aplikacija mora tudi omogočati poljubno konfiguracijo minutnega intervala in višine vrstice, ki predstavlja ta minutni interval.

Vsaka kolona vsebuje glavo, kjer je prikazan datum. Ta glava mora ostati vidna in vedno na vrhu, ko uporabnik naredi vertikalni poteg. V primeru horizontalnega potega pa se mora premakniti skupaj s kolono.

Pri horizontalnem potegu se na zaslonu prikažejo novi dnevi na koledarju. Zaradi optimizacije zmogljivosti se ti dnevi nalagajo in brišejo po potrebi. Vsak dan potencialno vsebuje zelo veliko DOM elementov za prikaz, vključno s slikami. V koledarju se lahko premikamo v časovno neskončnost. Implementacija, kjer bi se vsi dnevi naložili ob startu aplikacije, tako ne pride v poštev. Koliko dni naj bo naloženih vnaprej smo določili s testiranjem na fizičnih napravah. Izkazalo se je, da zadoščata dve koloni tudi pri večjih in hitrejših potegih.

4.4.3 »Material Design« elementi

Pri grafičnem oblikovanju aplikacije smo se oprli na smernice »Material Design«, ki jih je razvil Google. Te smernice so bile posebej razvite za uporabo na mobilnih napravah in uporabljajo

nove tehnologije, ki so nam jih prinesle mobilne naprave. Poudarek je na pametnem strukturiranju sklopov v aplikaciji s pomočjo senčenja in animacij. Tako ima uporabnik občutek, da je aplikacija tridimenzionalna. Pri mnogo uporabniških kretnjah prekrijemo grafični element s prstom in ga tako za kratek čas ne vidimo. Zaradi tega so na primer klasične spremembe oblike pri pritiskih gumbov nepotrebne. Bolje je narediti animacijo na gumbu ali katerem koli drugem elementu, da uporabnik, po tem ko umakne prst vidi, da je aplikacija njegovo kretnjo registrirala.

Za čas dela te diplomske naloge je bila uporabljena knjižnica Sencha Touch 2.4, ki ni bila na podlagi teh oblikovnih smernic. Zato je bilo potrebno nekaj elementov prilagoditi. Ti elementi so gumb, potrditveno polje in polje za iskanje.

4.4.3.1 FRI.ux.MaterialButton

Standardni HTML gumbi imajo dve grafični stanji: stanje, ko gumb ni pritisnjen in stanje ko je. Na mobilnih napravah to ni optimalno, saj pri manjših gumbih uporabnik lahko ne dobi potrditve, da je bil pritisk na gumb izveden. Zato je potrebno uporabniško izkušnjo nadgraditi. Potrditev pritiska mora ostati vidna še nekaj časa, ko uporabnik že umakne gumb.

»Material design« smernice tudi priporočajo uporabo gumbov, ki nimajo obrobe v primerih, kjer že imamo en dvignjen nivo uporabniškega vmesnika. To se pri naši aplikaciji vidi pri oknih s podrobnostmi učne ure na koledarju.

Standardne Sencha Touch gumbe smo nadomestili z gumbi iz knjižnice Waves [64]. Knjižnica omogoča gumbe z robovi in tudi brez. Gumbi ob pritisku prikažejo animacijo, ki je podobna valovanju vode.

4.4.3.2 FRI.ux.MaterialCheckbox

FRI.ux.MaterialCheckbox je element, ki ga uporabljamo namesto klasičnega potrditvenega polja. Neoznačeno polje ima obliko kvadrata. Ko polje označimo, se izvede animacija, kjer se ta kvadrat zavrti in spremeni v kljukico. Ta element je v aplikaciji uporabljen pri pogledu »indeks« kjer lahko uporabnik določi, kateri predmeti naj bodo prikazani.

4.4.3.3 FRI.ux.StaffSearchButton

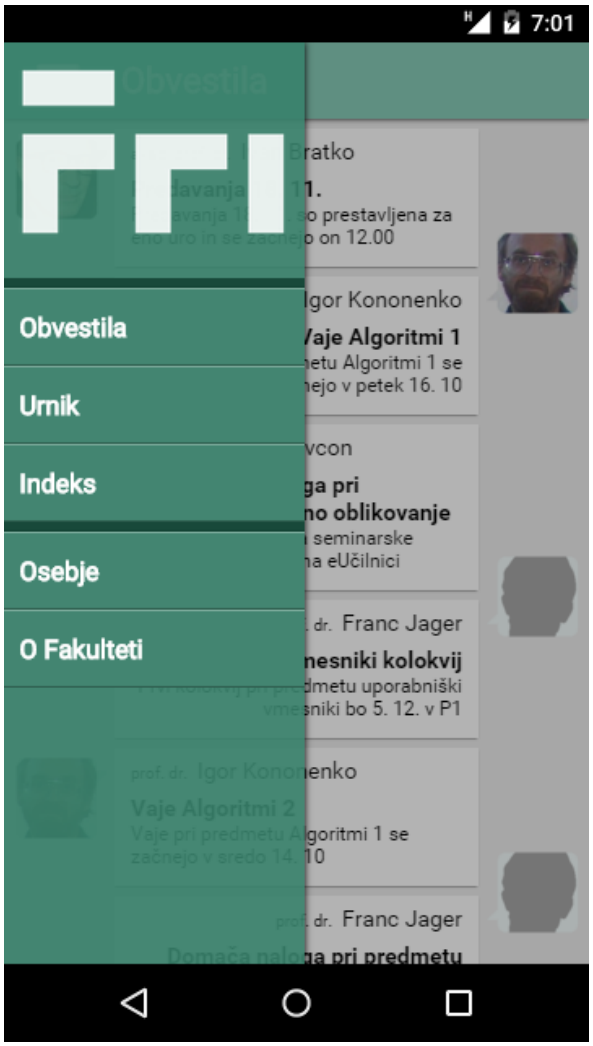
FRI.ux.MaterialCheckbox je element, ki je uporabnikom na voljo pri pogledu »Osebje«. Polje je v obliki gumba dokler uporabnik ne pritisne nanj. Ob pritisku pa se gumb spremeni v polje za vnos teksta.

4.5 Umestitev zunanjih komponent v aplikacijo

Zunanje komponente kot so knjižnice Lawnchair [65], Lodash [66], Waves [64] je bilo potrebno prilagoditi in jih spremeniti v Sencha Touch razrede. Sencha Touch CMD orodja omogočajo dva načina izdelave aplikacije. Razvojno verzijo in produkcijsko verzijo. Razvojna verzija omogoča uporabo zunanjih orodij brez posebne prilagoditve. Produkcijska verzija pa tega ne omogoča saj združi vse datoteke z Javascript kodo v eno samo datoteko.

4.6 Predstavitev končne aplikacije

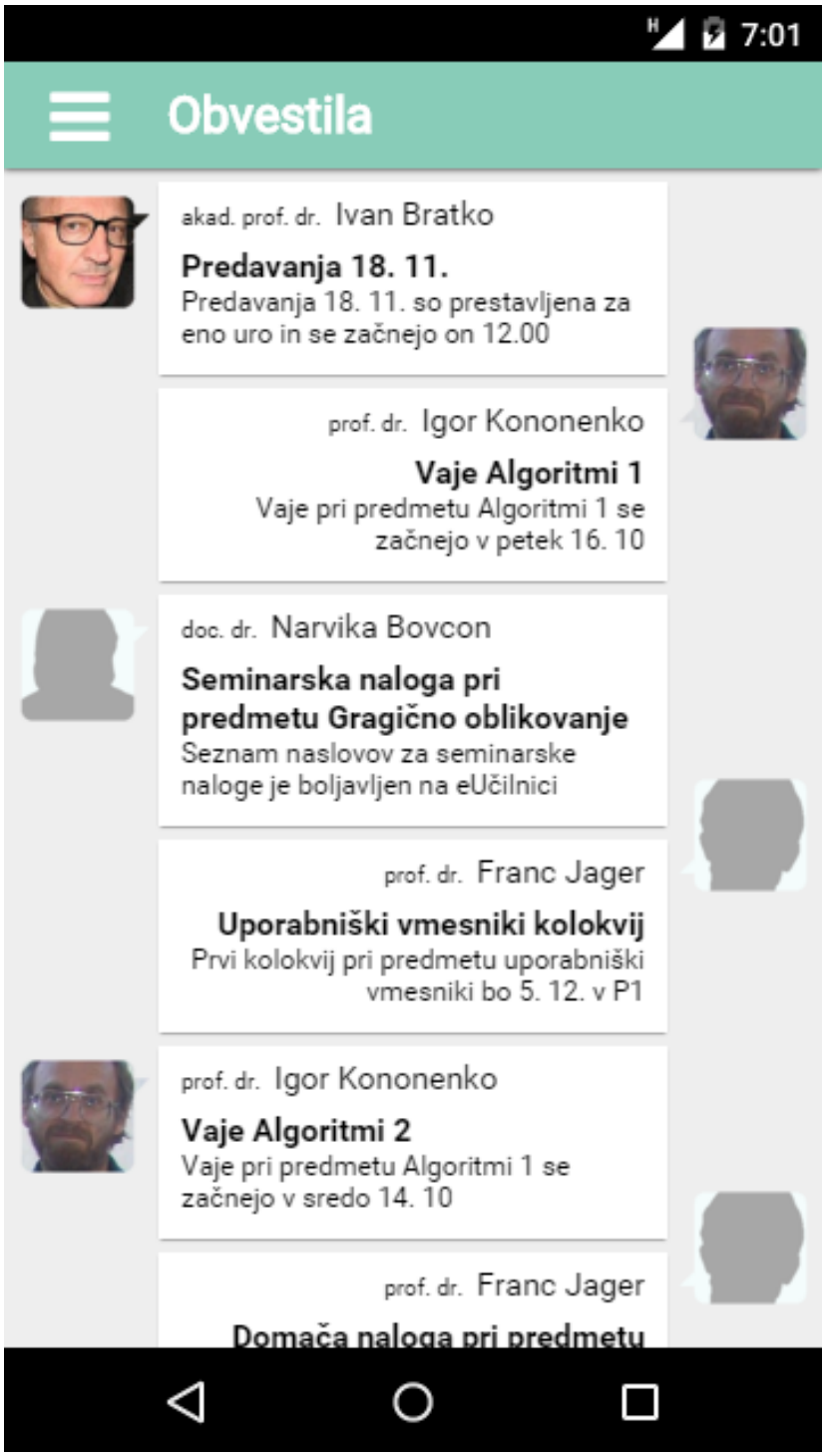
Končna mobilna aplikacija vsebuje naslednje funkcionalnosti, ki so konceptualno razdeljene v naslednje vizualne sklope: obvestila, urnik, indeks, osebje in o fakulteti. Do vsakega pogleda uporabnik lahko dostopa preko menija, ki ga aktivira s pritiskom na gumb v zgornjem levem kotu.



Slika 4.4: Glavni meni mobilne aplikacije

4.6.1 Obvestila

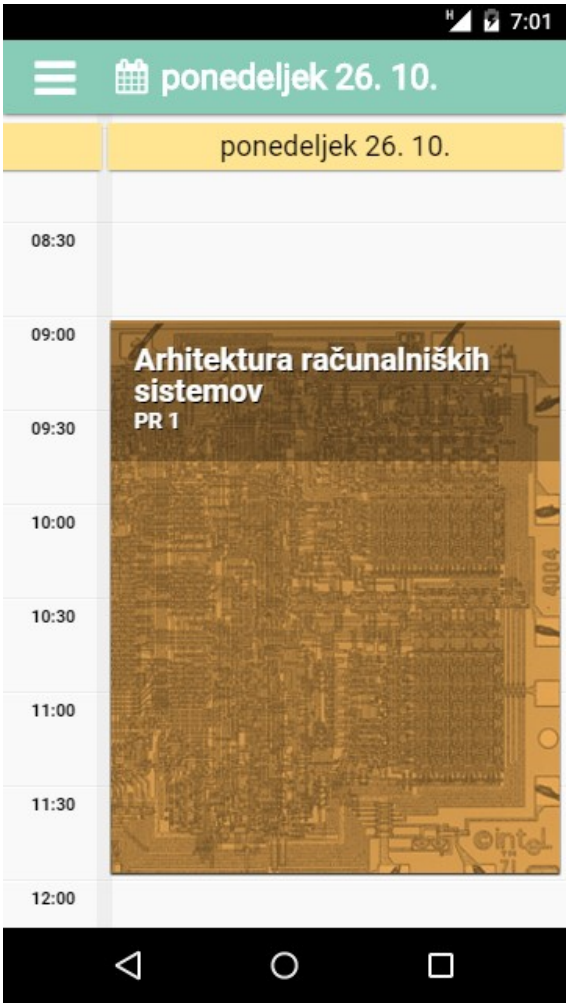
Obvestila so prvi pogled, ki ga uporabnik vidi, ko se prižge aplikacija. Na prvo mesto so postavljena zaradi pogostega posodabljanja in pomembnosti. Obvestila lahko dodaja samo osebje.



Slika 4.5: Pogled »Obvestila«

4.6.2 Urnik

Urnik je pogled, ki uporabniku omogoča pregled nad predmeti, ki jih opravlja. Pogled se prilagaja velikosti zaslona. Na večjih zaslonih je hkrati prikazanih več dni, na manjših pa lahko tudi samo eden. S pritiskom na naslov predmeta se uporabniku prikaže okno z dodatnimi informacijami o predmetu. V istem oknu pa je tudi možnost za razpravo o specifični učni uri, kjer lahko osebje napiše dodatne informacije oz. opiše kaj se bo na učni uri obravnavalo.

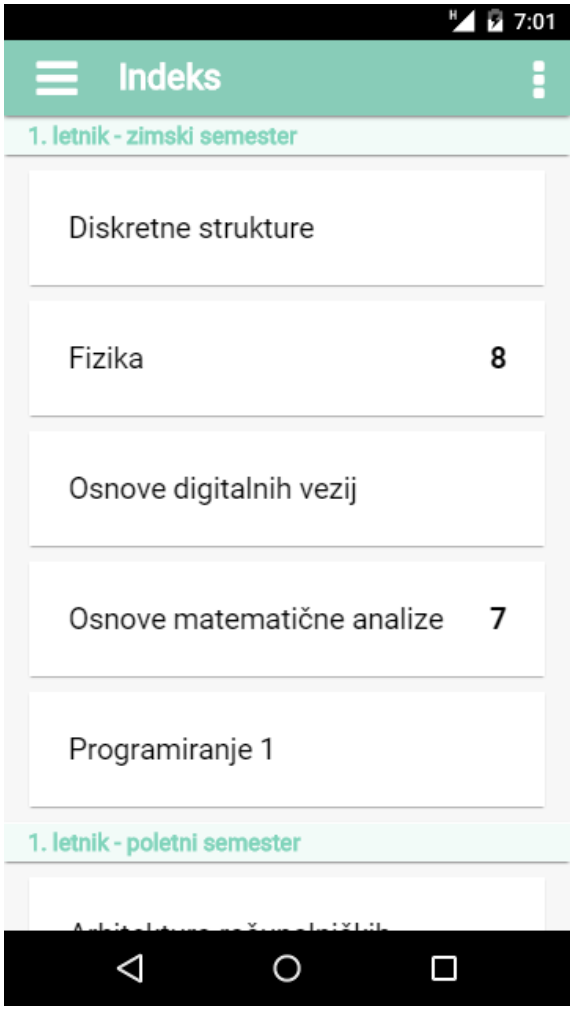


Slika 4.6: Pogled »Urnik«

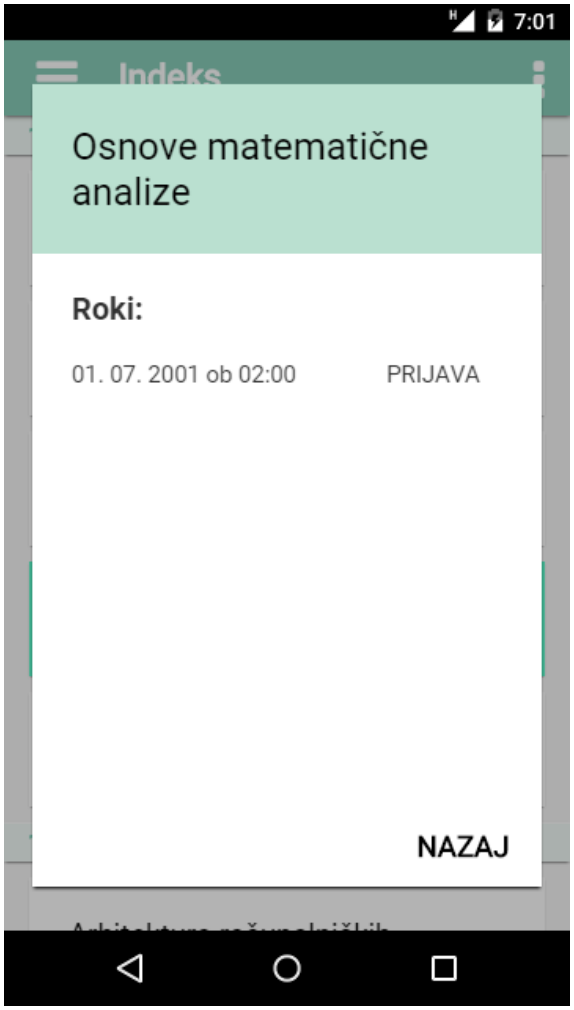
Slika 4.7: Okno z dodatnimi informacijami o predmetu

4.6.3 Indeks

Pogled »Indeks« omogoča uporabniku pregled predmetov, ki jih opravlja. Predmeti, ki jih je že opravil imajo poleg naslova napisano tudi oceno. S klikom na predmet se uporabniku prikaže novo okno, kjer so vidni možni izpitni roki z gumbom za prijavo ali odjavo. V zgornjem desnem kotu je gumb, kjer lahko uporabnik dostopa do filtrov. S filtri lahko prikaže le opravljene ali neopravljene predmete in le določene letnike.



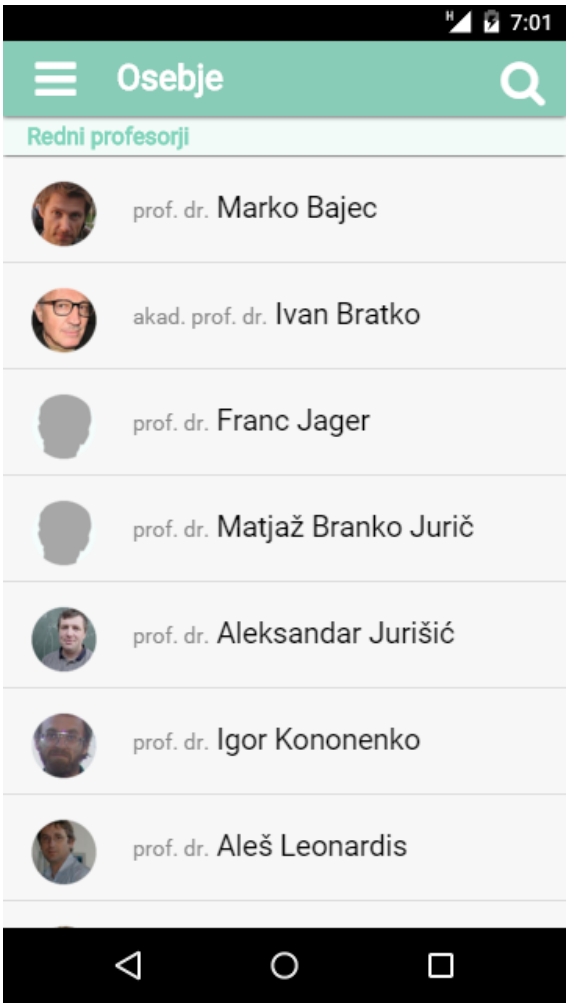
Slika 4.8: Pogled »Indeks«



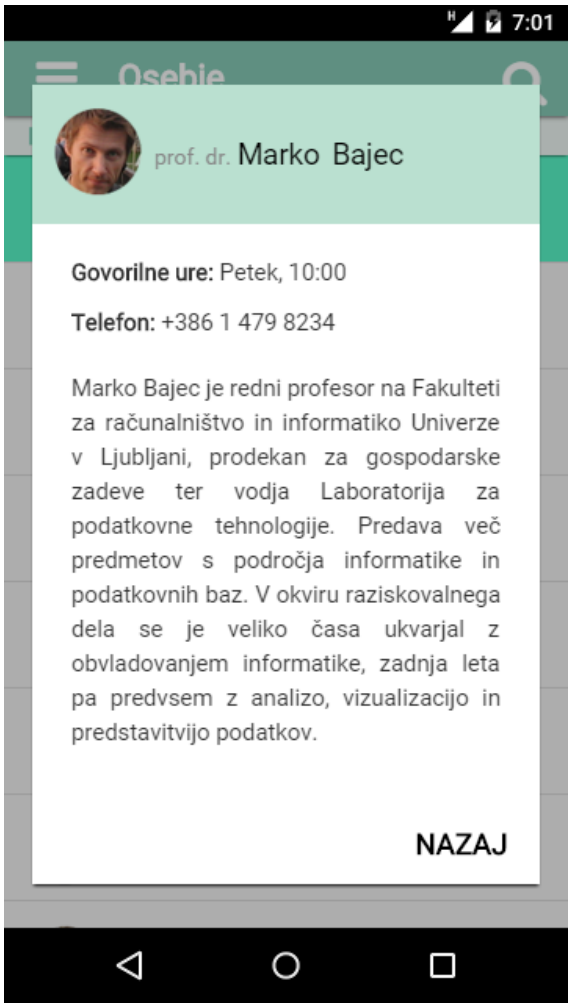
Slika 4.9: Okno za prijavo na izpit

4.6.4 Osebje

Pogled »Osebje« vsebuje seznam osebja fakultete. S pritiskom na osebo se odpre okno s kratko predstavitvijo, informacijami o govorilnih urah in kontaktom te osebe. V zgornjem desnem kotu, je gumb za iskanje s katerim lahko uporabnik poišče osebe z vpisom imena ali priimka.



Slika 4.10: Pogled »Osebje«



Slika 4.11: Okno z dodatnimi informacijami o osebi

4.6.5 O fakulteti

Zadnji pogled »O fakulteti« vsebuje splošne podatke o inštituciji in uporabne kontakte.

Slika 4.12: Pogled »O fakulteti«

Poglavje 5 Sklepne ugotovitve

V diplomskem delu smo dosegli vse zastavljene cilje. V drugem poglavju smo prvi zastavljen cilj v diplomskem delu in opravili raziskavo trga mobilnih aplikacij za pomoč pri študiju. Ugotovili smo, da so take aplikacije že na trgu in da nekatere, generične in specifične, dosegajo dobre rezultate pri prenosih, kot tudi pri uporabnosti in zadovoljstvu uporabnikov. Ugotovili smo tudi, da veliko klasičnih študijskih informacijskih sistemov podpira razširitvene rešitve, ki vključujejo mobilne aplikacije.

Drugi cilj smo izpolnili v tretjem poglavju, kjer smo analizirali programska orodja in strežniško infrastrukturo. Pri planiranju razvoja aplikacije smo se osredotočili na spletne tehnologije v oblaku in zastavili zaledje za aplikacijo na AWS platformi. Zaledje deluje neodvisno od študijskih informacijskih sistemov in omogoča optimizirano komunikacijo med strežnikom in množico aplikacij. Ta rešitev nam omogoča kasnejšo integracijo z različnimi informacijskimi sistemi, hkrati pa zagotavlja robustno platformo za nadaljnji razvoj.

Zadnji cilj diplomske naloge smo uresničili z razvojem mobilne aplikacije, ki študentom in pedagogom omogoča dostop do ključnih funkcionalnosti, kot jim jih nudijo študijski informacijski sistemi. Ta razvoj smo opisali v četrtem poglavju. Aplikacija je bila narejena po najnovejših standardih, ki zagotavljajo uporabnikom najlažjo uporabniško izkušnjo. Izbrane so bile funkcionalnosti, ki so po našem mišljenju in raziskavi v analizi trga najbolj primerne. To so: novice, urnik, indeks, pregled osebja in splošne informacije o fakulteti.

To aplikacijo lahko uporabimo za nadaljnji razvoj in raziskave koristnosti in dodane vrednosti takšne aplikacije. Primerna je za izvajanje uporabniških študij na študentih in osebju fakultete. Z dodatnim vložkom v integracijo pa jo lahko tudi uporabimo kot produkcijsko aplikacijo kot dopolnilo k že obstoječima študijskima informacijskima sistemoma eŠtudent in eUčilnica.

Trenutno so omejitve aplikacije, da je omejena na delovanje v svojem zaprtem okolju in ni integrirana v noben študijski informacijski sistem, hkrati ne uporablja sej in nima izdelanega vmesnika za prijavo in registracijo. To so nadgradnje, ki bi jih bilo potrebno izvesti ob umestitvi v študijski informacijski sistem. Kot del prvega prototipa jih ni bilo smiselno razvijati zaradi specifičnosti študijskega informacijskega sistema na FRI in pravil o varnosti osebnih podatkov.

Literatura

[1] Edutechnica, „LMS Data: 3rd Annual Update.," 10 10 2015. [Elektronski]. Dostopno na: http://edutechnica.com/2015/10/10/lms-data-3rd-annual-update/. [Poskus dostopa 17 11 2015].

[2] [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.blackboard.android.

[3] [Elektronski]. Dostopno na: http://eduappcenter.com/.

[4] [Elektronski]. Dostopno na: https://appfinder.brightspace.com.

[5] [Elektronski]. Dostopno na: http://www.dublabs.com/schools/.

[6] P. Thiel in B. Masters, „All happy companies are different," v *Zero to One: Notes on Startups, or How to Build the Future*, Crown Business, 2014, pp. 23 -25.

[7] D. Schaffhauser, „In Yearly LMS Count Canvas Gains, Blackboard Learn Loses; Campus Technology," 14 10 2014. [Elektronski]. Dostopno na: https://campustechnology.com/articles/2014/10/14/in-yearly-lms-count-canvas-gains-blackboard-learn-loses.aspx?admgarea=News.

[8] IDC, „IDC: Smartphone OS Market Share 2015, 2014, 2013, and 2012," [Elektronski]. Dostopno na: http://www.idc.com/prodserv/smartphone-os-market-share.jsp..

[9] U. o. P. Mobile. [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=edu.apollogrp.android.classroom.activity.

[10] „Ashford University Mobile," [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=edu.ashford.talon.

[11] „Christ University Student App," [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.yaragos.christuniversity.

[12] „Monash University," [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.monashuniversity.monashuniversity.

[13] „Kingston University,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.ombiel.campusm.kingston.

[14] „Newcastle University,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=uk.ac.ncl.m.

[15] „University Bordeaux Schedule,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.bordeaux1.emplois.

[16] „Monroe Community College,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.dub.app.MCC.

[17] „Calvin College,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=edu.calvin.calvinmobile.

[18] „Hathershaw College,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.appscentral.thehathershawcollege.

[19] „Emerson College,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=edu.emerson.m.

[20] „St Charles Sixth Form College,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.appscentral.stcharlescatholicsixthformcollege.

[21] „Visit Fisher,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=edu.osu.fisher.visit_fisher.

[22] „Sensor Tower - Mobile App Store Marketing Intelligence,“ [Elektronski]. Dostopno na: https://sensortower.com.

[23] „UniOnline for KF Graz,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/unionline-for-kf-graz/id1028625440?mt=8.

[24] „OOHLALA - Campus App with Events Calendar, Class Schedule and Friends' Timetable,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/oohlala-campus-app-events/id389713598?mt=8.

[25] „iWestminster - University of Westminster Mobile Application,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/iwestminster-university-westminster/id410340418?mt=8.

[26] „The University of Texas At Austin,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/university-texas-at-austin/id347883008?mt=8.

[27] „University of Phoenix Mobile,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/university-of-phoenix-mobile/id429092408?mt=8.

[28] „University of Sussex – SussexMobile Application,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/university-sussex-sussexmobile/id385527472?mt=8.

[29] „Timetable,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.gabrielittner.timetable.

[30] „My Study Life,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.virblue.mystudylife.

[31] „Google Classroom,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.google.android.apps.classroom.

[32] „myHomework Student Planner,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.myhomeowork.

[33] „Student Agenda,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.clawdyvan.agendadigitalaluno.

[34] „Handy Timetable,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=com.newbitmobile.handytimetable.

[35] „My Class Schedule: Timetable,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=de.rakuun.MyClassSchedule.free.

[36] „Students - Timetable,“ [Elektronski]. Dostopno na: https://play.google.com/store/apps/details?id=crazy.students.student.

[37] „myHomework Student Planner,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/myhomework-student-planner/id303490844?mt=8.

[38] „Pocket Schedule - Class Schedule, Homework Planner & School Organizer,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/pocket-schedule-class-schedule/id421102532?mt=8.

[39] „ClassManager - Student School Schedule Planner Tracker & Assignment List Homework Organizer App,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/classmanager-student-school/id529458056?mt=8.

[40] „Courses - Grade Tracker and Student Report Card,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/courses-grade-tracker-student/id820707272?mt=8.

[41] „Student Calendar & Homework Assignment Planner By Planneroo™,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/student-calendar-homework/id704198994?mt=8.

[42] „Timetable - Weekly schedule planner for Universities, School and Colleges,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/timetable-weekly-schedule/id916497066?mt=8.

[43] „Homework Suite- Student Planner, Class Schedule & School Calendar with Timetable,“ [Elektronski]. Dostopno na: https://itunes.apple.com/si/app/homework-suite-student-planner/id1031496229?mt=8.

[44] IIBA, "MoSCoW Analysis (6.1.5.2)". A Guide to the Business Analysis Body of Knowledge (2 ed.)., K. Brennan, Ured., International Institute of Business Analysis, 2009.

[45] „MoSCoW Prioritisation,“ [Elektronski]. Dostopno na: http://www.dsdm.org/content/10-moscow-prioritisation.

[46] „Crosswalk,“ [Elektronski]. Dostopno na: https://crosswalk-project.org/.

[47] „Sencha Touch,“ [Elektronski]. Dostopno na: https://www.sencha.com/products/touch/.

[48] Sencha, „Licensing – Sencha,“ [Elektronski]. Dostopno na: https://www.sencha.com/legal/.

[49] „Ionic Framework,“ [Elektronski]. Dostopno na: http://ionicframework.com/.

[50] „React,“ [Elektronski]. Dostopno na: https://facebook.github.io/react/.

[51] „Polymer,“ [Elektronski]. Dostopno na: https://www.polymer-project.org.

[52] „Cordova,“ [Elektronski]. Dostopno na: https://cordova.apache.org/.

[53] „Phonegap,“ [Elektronski]. Dostopno na: http://phonegap.com/.

[54] PhoneGap, „PhoneGap, Cordova, and what’s in a name?,“ [Elektronski]. Dostopno na: http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/.

[55] „Closure Compiler,“ [Elektronski]. Dostopno na: https://developers.google.com/closure/compiler/.

[56] „JScrambler,“ [Elektronski]. Dostopno na: https://jscrambler.com/en/.

[57] „Node.js,“ [Elektronski]. Dostopno na: https://nodejs.org.

[58] „Express,“ [Elektronski]. Dostopno na: http://expressjs.com/.

[59] „Vogels,“ [Elektronski]. Dostopno na: https://github.com/ryanfitz/vogels.

[60] „node-imagemagick-native,“ [Elektronski]. Dostopno na: https://github.com/elad/node-imagemagick-native.

[61] „ImageMagick,“ [Elektronski]. Dostopno na: http://www.imagemagick.org/Magick++/.

[62] „Git,“ [Elektronski]. Dostopno na: https://git-scm.com/.

[63] „JSONDiffPatch,“ [Elektronski]. Dostopno na: https://github.com/benjamine/jsondiffpatch.

[64] „Waves,“ [Elektronski]. Dostopno na: http://fian.my.id/Waves/.

[65] „Lawnchair,“ [Elektronski]. Dostopno na: htttp://brian.io/lawnchair/.

[66] „Lodash,“ [Elektronski]. Dostopno na: http://lodash.com.